A SENSITIVITY ANALYSIS OF THE PRICE S AND SYSTEM-3 SOFTWARE COST ESTIMATING MODELS

THESIS

Christina E. Voss
Captain, USAF

AFIT/GCA/LSQ/88D-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

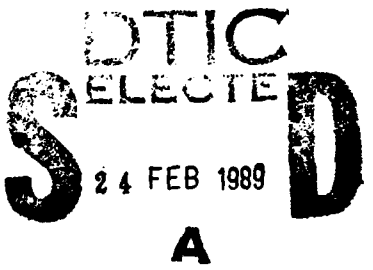
AFIT/GCA/LSQ/88D-11

A SENSITIVITY ANALYSIS OF THE PRICE S AND SYSTEM-3 SOFTWARE COST ESTIMATING MODELS

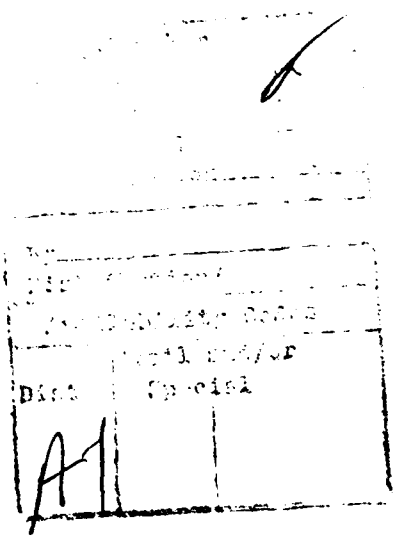
THESIS

Christina E. Voss
Captain, USAF

AFIT/GCA/LSQ/88D-11



The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information is contained therein. Furthermore, the views expressed in the document are those of the author and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the United States Air Force, or the Department of Defense.



AFIT/GCA/LSQ/88D-11

A SENSITIVITY ANALYSIS OF THE PRICE S AND SYSTEM-3 SOFTWARE COST ESTIMATING MODELS

THESIS

Presented to the Faculty of the School of Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Cost Analysis

Christina E. Voss, B.A.
Captain, USAF

December 1988

Acknowledgements

During the accomplishment of this research, I received help from a number of people. I owe thanks to my thesis advisor, Mr Rich Murphy, for giving me guidance and for having faith in me through to the end. Capt Tony Gallo, ASD/ACCR, was instrumental as my point of contact in ASD, and also gave me much help with the PRICE S model. I'd like to thank Mr Jim Otte and Ms Carol Fugate of PRICE Systems, for the great assistance they provided on the operation of the PRICE S model. Also, I'd like to thank Mr Wayne Stanley of CEI, for all of his valuable help on the System-3 model. In addition, I want to thank CEI for the unfunded use of their model. Finally, I wish to thank my fellow classmates, especially Capt Rod Troyanowski, for their assistance in turning out the final product.

Christina E. Voss

Table of Contents

List of Tables

Abstract

The purpose of this ~~research~~ effort was to perform an analysis on the PRICE S and System-3 software cost estimating models, in order to determine the domains of consistency for four software development efforts. That is, to determine the sets of inputs for which the two models provide approximately equal estimates.

It was necessary to examine the inputs required to run PRICE S, to determine reasonable values for those inputs to describe the four software development efforts of increasing complexity, and to run the PRICE S model to get the estimates for performing those efforts. The PRICE S estimates were used as the baseline for the sensitivity analyses on the System-3 variables.

The System-3 variables were examined and compared to the PRICE S variables. There were three types of relationships between the variables. There were some System-3 inputs that had one-to-one correspondence with PRICE S inputs, some that when combined with other System-3 inputs corresponded to a single PRICE S variable, and a few that had no PRICE S equivalent.

The sensitivity analysis on the System-3 variables was only applied to the ten variables that related to the PRICE S variable CPLX1 and the three variables that related to the PRICE S variable UTIL. The sensitivity analyses resulted in

the determination of the domains of consistency for the four software efforts. Because of an infinite number of possible variable combinations, no one specific domain of consistency could be specified. Rather, the domains of consistency were represented by changes allowed in the variables, individually and as a group.

A SENSITIVITY ANALYSIS OF THE PRICE S AND SYSTEM-3 SOFTWARE COST ESTIMATING MODELS

I. Introduction

General Issue

The task of estimating software costs is a difficult one but one of vital importance if the cost estimate of a weapon system, as a whole, is to be useful to the decision maker. Throughout this study, the term software will refer to software that is an integral component of weapon systems. That is, it is "physically a part of, dedicated to, or essential in real time performance of the mission of a weapon system" (5:4). Previously, it was known as embedded software but is now called mission critical computer software (5:4). This software represents significant component costs of complex avionics systems managed by the Air Force. Cost estimates are used during the resource allocation process to determine where and how scarce funds should be spent. Cost estimates are vital to capital budgeting decisions, source selection decisions, project scheduling, system design trade-offs, and performance evaluation (20:2-3). Evaluating the performance of contracts is the program manager's job. If software costs

cannot be adequately estimated, the program manager's job is made more difficult since he does not have a reasonable estimate against which to control and manage program costs. This dilemma can be resolved if cost analysts can provide the program manager with these reasonable software cost estimates. To do so, cost analysts must have software cost estimating models which can consistently provide reasonable software cost estimates. The problem arises in deciding which software cost estimating model the cost analyst should use.

Specific Problem

A number of software cost estimating models and techniques are currently used by industry to determine proposed costs for systems under procurement, and by government to evaluate and determine the development costs of mission critical computer software. These estimates are used by government managers to make critical decisions during the development life cycle. As mentioned before, the problem arises in deciding which model should be used for determining the cost of software crucial to the operation of complex weapon systems. A typical method for comparing models is to see which one provides the "best estimate" when compared to actual costs. Another method for comparing software cost estimating models is to analyze the effects of different values for the models' inputs on the resulting estimates. If the models provide equal estimates, they can

be analyzed further to determine by how much input values may vary, yet still provide approximately equal results in terms of the estimated costs of mission critical computer software. In other words, given the estimate from one model, how much may the other model's input variables change without exceeding acceptable limits around that estimate? On the other hand, if the models do not provide equal estimates for comparable software, probable causes need to be identified. The ranges of input values for which the models give approximately equal results, will be called the domain of consistency. That is, for that particular range, the models are very compatible. On the other hand, the domain of inconsistency would represent that range for which the models give different estimates.

Research Objectives

The purpose of this research is to perform an analysis of two of the software cost estimating models (PRICE S and System-3) currently used by industry and government, in order to achieve the following research objectives:

1) Determine the cost estimate produced by PRICE S for each of four software development efforts representing simple, moderate, complex, and very complex projects.

2) Determine the domain of consistency, both individually and as a group, for selected inputs to the System-3 model when the criteria for consistency are based on the PRICE S model.

3) Determine if each domain of consistency includes input values that could reasonably be used to desribe the corresponding software project.

Investigative Questions

During the course of this research, the following questions will be considered for research objective #1:

1) What inputs are required to run the PRICE S model?

2) What would be reasonable values for these inputs to describe four software development efforts representing increasingly complex levels of performance?

3) What cost estimates would PRICE S produce for each of the four software development efforts?

The following questions will be considered for research objective #2:

1) What inputs are required to run System-3?

2) What is the relationship between the System-3 inputs and the PRICE S inputs?

a) What inputs have a one-to-one correspondence between the two models?

b) What PRICE S inputs correspond to multiple System-3 inputs?

c) What inputs are unique to each model?

3) For each software development effort, which System-3 inputs should be set to specific values and which inputs should be allowed to vary?

4) What is the domain of consistency for those variables that are allowed to vary?

The following question will be considered for research objective #3:

1) Are there any sets of inputs in the domain of consistency that could describe a software development effort equivalent to the effort that generated the PRICE S estimate? If so, what are they?

Scope of Research Effort

There are at least ten major software cost estimating models (1). However, many are simply modifications of an established model to adapt it to a specific application. Due to the amount of time required for a sensitivity analysis, and given a limited time period, only two models will be evaluated. These models are the PRICE S model and the System-3 model. PRICE S is a trademark of PRICE Systems (General Electric Co.), while System-3 is a trademark of Computer Economics, Inc (CEI).

Selection of these two models was determined by the application of two basic criteria. The first criterion was that the model be used extensively by the Air Force and the second criterion was that the models not be related to each other. By related, it is meant that the models are closely tied together in that one is a modification of the other or they are based on similar ideas.

For purposes of this study, a typical Air Force organization will be used as a point of reference. The product divisions of Air Force Systems Command (AFSC) all use software cost estimating models when estimating the costs of various weapon systems. Armament Division (AD) at Eglin AFB, Electronic Systems Division (ESD) at Hanscom AFB, Space Division (SD) at Los Angeles AFB, and the Aeronautical Systems Division (ASD) at Wright-Patterson AFB use the PRICE S and System-3 models, among others. More

specifically, ASD uses both PRICE S and System-3 for estimating software development costs for avionics systems procured by its System Program Offices (SPOs). Thus, the first criterion is met. The selection of these models is further justified by the fact that the System-3 model is currently competing with the PRICE S model to be the primary software cost estimating model used by ASD.

The two models will be discussed in more detail in Chapters 2 and 3; but, for now, a brief description is necessary to show how the second criterion is met. Although both PRICE S and System-3 are proprietary models, basic information about their processing is known. PRICE S is based on managerial perspectives, using the Delphi method. After the fact, a regression-based model was formulated to validate what the Delphi method had already established (7). On the other hand, System-3 is based primarily on the Rayleigh-Norden curve (7). It also incorporates features of other models such as the COCOMO model and the SLIM model. Therefore, PRICE S and System-3 are distinctive models; they are not related. Thus, the second criterion is met. In summary, both the PRICE S model and the System-3 model satisfy the selection criteria.

Justification

The Comptroller Office of the Aeronautical Systems Division, ASD/ACC, has expressed interest in research in this area and will provide assistance in accomplishing the

research. This research will determine the conditions, that is, the domain of consistency, for which the PRICE S model and the System-3 model produce nearly equal estimates. When inputs fall within the domain of consistency, the least expensive model to run should be used.

Limitations of Research Effort

This research will not attempt to say anything about which model to use when the inputs fall outside the domain of consistency, that is, the domain of inconsistency. In that case, it would be important to run both models and attempt to reconcile the estimates. Additionally, this study will limit the estimating process to the project development phase of the software life cycle. That phase closely coincides with the program development function of the System Program Offices.

II. Literature Review

Software cost estimating capabilities have been the subject of much research in recent years. Software costs cannot be pinpointed exactly, mainly because software is still relatively new to the world of cost estimation. Software was previously regarded as an element of hardware and as such, its cost was included in the cost of the hardware. However, in recent years, software has become the item of procurement rather than hardware. "The Department of Defense estimates that by 1990, 85% of the budget spent on computer related acquisitions will be spent on software" (23:6). This would imply that software costs need to be separated from hardware costs and estimated separately. This literature review will cover three areas concerning the estimation of software costs. The first area covers the inexact nature of software and the difficulty associated with developing software cost estimates. The second area looks at some of the models developed for the purpose of estimating software costs, and the third area examines various studies done on software cost estimating methodologies and the models themselves.

Software Costs

The estimation of software costs is a difficult process and one which gives somewhat inexact estimates. In fact, current software cost estimating models usually predict

software costs within 20% of the actual costs, but that only occurs in approximately 70% of their predictions (1:32). According to Dr Barry W. Boehm, the main reasons for the inexactness of software cost estimates are as follows:

1) Source instructions are not a uniform commodity, nor are they the essence of the desired product.

2) Software requires the creativity and cooperation of human beings, whose individual and group behavior is generally hard to predict.

3) Software has a much smaller base of relevant quantitative historical experience, and it is hard to add to the base by performing small controlled experiments [1:32].

Others have made similar conclusions. Mitre Corporation conducted the 1975 DOD Weapon System Software Acquisition and Management Study and found that:

> Meaningful cost information was not generally available. This was apparently due to lack of common definitions for the components of software costs, to regulations not requiring software to be broken out and maintained separately from hardware, and to a lack of detailed historical cost records. It was also noted that cost information was rarely correlated with technical information for management purposes. Future efforts to determine the cost of software in weapon systems should include (start with) the development of a management cost model and agreement on its content. [15:6].

Although it is difficult to predict software costs at the beginning of a program, it is generally acknowledged that the estimates improve over time. In fact, "most cost models assume that their predictions will increase in accuracy as the software development cycle progresses and the software product becomes better defined" (13:273). As the product is better defined, it is easier to break a large

software system into smaller modules. Once the software is broken down into programs and subprograms, these smaller modules can be estimated based on similar developments or experiences (6:18). The ability to control software during the development phase is vital to the breakdown of software. If the software cannot be controlled, it cannot be adequately defined for estimation purposes. Established policies to control software during the development phase make software cost estimation an easier task (14:15).

It is also difficult to estimate software costs because there are a number of factors which cause software costs to differ. Some of these factors are laid out in the PRICE S Reference Manual (18). Product characteristics, project constraints, technical capability, and management factors all cause variability in software costs. Product characteristics such as size, type of software, specifications, processor loading, and interfaces (between hardware and software) all contribute to cost variability. Schedule, manpower, inventory (design/code), changing requirements, and multi-team coordination are subfactors of the project constraints. The components of technical capability that can cause software costs to differ are personnel (skill/experience levels), product familiarity, and development aids/facilities. Finally, management factors affecting software costs are rates (labor/overhead), procedures/habits, and economics/inflation.

Two of the factors mentioned above, size and schedule, appear to have particularly important impacts on the cost of software development. Therefore, they are worthy of more attention and will be covered in more detail. Size is a key input factor for determining software costs. It is represented by the number of lines of code, typically source lines of code. The main difficulty here is estimating the number of lines of code needed to satisfy the software requirements. Since size is so difficult to estimate, it is only natural that the related costs are also difficult to estimate. Size can be estimated by a number of techniques. Currently, there are five main ways of estimating software size (7). The first way is through analogy, that is, comparing the current software program to similar programs. Secondly, a regression-based technique involves the development of a sizing equation through the regression analysis of historical data (7). The third technique uses a "survey of one or more experts who can estimate size", thus giving an expert judgment (7). Next, function point models base the estimation on five factors: inputs, outputs, inquiries, master files, and interfaces. Finally, parametric models estimate size from values of multiple program attributes (7). A sixth technique to estimate size is merely a composite of any of the other five techniques.

Schedule is often represented in terms of man-months, which when converted to dollars, greatly affects software

development costs. Schedules, in terms of man-months, must be realistic if reasonable software development cost estimates are to be obtained. Frederick Brooks discusses the man-month in great detail in his essay, "The Mythical Man-Month." He contends that it is wrong to assume "that men and months are interchangeable" (2:16). If a task requires communication among team members, men cannot be exchanged for months. That is, the addition of more men will not decrease the number of months required to complete the task. The use of more manpower typically requires more communication, which in turn, requires more time. An oversimplification of this idea is represented by Brooks' Law: "Adding manpower to a late software project makes it later" (2:25). This law can be avoided only when the tasks are truly independent and no communication is necessary.

An often overlooked factor that affects software costs is software quality. How does one measure quality, and possibly put a pricetag on it? Finding the cost of software quality depends on a successful quality program. Charles Holloker identified three vital elements that lead to a successful quality program. Those elements are as follows:

1) The ability of the company's accounting system to present appropriate cost summaries.

2) The capability of identifying all costs associated with the quality function.

3) The quality function's ability to control quality costs. [11:224].

Factors which may have a great impact on software cost estimates in the future include the increased use of the Ada programming language, new development and support concepts, and artificial intelligence (8:31-32). These items are covered only briefly. Models will need to be adapted as Ada becomes a mature programming language. Models will also need to be updated to accomodate new ways of handling the software development process. Finally, the advent of artificial intelligence (AI) will require that AI software management receive more attention because of the "software-intensive" nature of AI (8:32). And, once again, models must be adapted to account for this new technology.

Software Cost Estimating Models

The most commonly used software cost estimating models are parametric models. That is, software costs are estimated on the basis of design characteristics of the software program. There are three advantages in using parametric models. "They are usually fast and easy to use, require little detailed information, and capture system-level costs" (8:6). Parametric models are not without their disadvantages: there is a degree of instability, they are not always accurate, and the future is sometimes predicted using the past (7). However, the advantages outweigh the disadvantages. The requirement for little detailed information makes parametric models attractive to the DOD because they are "useful in early phases of a software

development program" and require little time to implement (8:6). It is appropriate now to discuss some of the currently available models that estimate software costs. Although there are quite a few models, only five will be discussed here to give a flavor of factors used by various models. The first three models discussed will be covered only briefly, while the PRICE S and System-3 models will be covered in more detail.

The Putnam SLIM model is a proprietary model with timeshare access for a fee. SLIM is an acronym for Software Life Cycle Model. It was initially developed by Lawrence H. Putnam, but has been marketed and modified by Quantitative Software Management Corporation (17). SLIM emphasizes the economics, trade-off opportunities, investment strategies, and financial control of software. It traces software costs from initiation (requirements analysis) through to operation (maintenance), with the intermediate stepping stones of definition (functional specification), design (construction specification), programming (coding), and testing (integration) (17:41). SLIM has three primary inputs: system size in number of developed source statements (work to be done), level or difficulty gradient (type of effort involved), and a technology factor (development environment) (24:A-73). Other inputs include labor and inflation rates, language, timing and memory utilization, use of modern programming practices, personnel qualification and/or

experience, and percentage of real time code (24:A-78, A-79).

COCOMO is a non-proprietary model which is well documented in Software Engineering Economics (1). COCOMO is short for the Constructive Cost Model, which was developed by Dr Barry Boehm. It estimates cost in terms of both man-months and schedule. It also accounts for new and modified design and code. The COCOMO model comes in three versions: basic, intermediate, and detailed. When dealing with embedded software under the basic version, it is necessary to use the COCOMO embedded mode. This is the most complex mode in the basic COCOMO model since it represents software that must operate within tight constraints (1:79). Both the intermediate and detailed COCOMO models use fifteen adjustment factors (cost drivers) to estimate software costs. These cost drivers are divided into product attributes, computer attributes, personnel attributes, and project attributes. Estimates are based on ratings given to the cost drivers listed in Table 1.

The FAST-E model is another proprietary model which can be accessed through timesharing. FAST-E is an acronym for the Freiman Analysis of Systems Techniques - Equipment model. It was developed by Frank Freiman (formerly of RCA), who was instrumental in developing the PRICE S model. The model is based upon the same principles as PRICE S, with some enhancements by Freiman (7). Like COCOMO, FAST-E

Table 1

COCOMO Cost Drivers (1:115-116)

Type of Attribute	Acronym	Variable
Product	RELY	Required Software Reliability
	DATA	Data Base Size
	CPLX	Product Complexity
Computer	TIME	Execution Time Constraint
	STOR	Main Storage Constraint
	VIRT	Virtual Machine Volatility
	TURN	Computer Turnaround Time
Personnel	ACAP	Analyst Capability
	AEXP	Applications Experience
	PCAP	Programmer Capability
	VEXP	Virtual Machine Experience
	LEXP	Programming Language Experience
Project	MODP	Modern Programming Practices
	TOOL	Use of Software Tools
	SCED	Required Development Schedule

estimates cost in terms of man-months, rather than dollars (8:11). It also has size as one of its primary inputs. Size can be a direct input or it can be computed internally by the model. Besides size, the other key input variables are as follows: 1) software type; 2) platform (operating environment); 3) engineering difficulty; 4) percentage of new design and new code; 5) percentage utilization of memory

or processing speed; 6) schedule start and end dates; and, 7) effort to integrate the specified program into a larger program (8:11).

PRICE S is also a proprietary model which can be accessed by computer timeshare, for a fee. It is one of five models in the PRICE family of models and concerns itself with the costs of software acquisition. PRICE is an acronym for Programmed Review of Information for Costing and Evaluation. The PRICE S model was developed by a number of PRICE Systems personnel. The PRICE models are based on managerial perspectives, using the Delphi method. After the fact, a regression-based model was formulated to validate what the Delphi method had already established (7). The PRICE parametric cost-modeling methodology is as follows:

> Interactive operation with conversational input and output.
>
> A parametric approach derived from experience and supported by empirical evidence.
>
> Efficient problem description with a small set of easily comprehended input factors.
>
> Internal self-checking to test the consistency of input data sets.
>
> Customizing flexibility so that the model may be adapted to local definitions and accounting procedures.
>
> Performance calibration that relates current estimates to actual achievements on prior projects [19:I-1].

This methodology does the following:

> provides a convenient way of reducing empirical data to a few principal variables which describe the

significant technological and cost differences between individual projects and organizations [19:I-2].

PRICE S estimates the manpower needed, the cost, or a typical schedule for computer software development. PRICE S considers seven main categories and one minor one as its principal inputs. The main categories are as follows:

1. Project Magnitude (How big?)
2. Program Application (What character?)
3. Level of New Design and Code (How much new work is needed?)
4. Productivity (Who will do the work?)
5. Utilization (What hardware constraints?)
6. Customer Specifications and Reliability Requirements (Where and how used?)
7. Development Environment (What complicating factors exist?) [19:I-7].

The other category is language type which specifies the source language to be used for the software development effort. The PRICE S input variables will be discussed in greater detail in Chapter III.

The PRICE S model was updated in November 1987. (For the benefit of those familiar with the old PRICE S model, RESO was replaced by PROFAC; CPLX was replaced by CPLXM, CPLX1, and CPLX2; and, LANG was added). The reader is referred to the PRICE S manual (19), for a complete description of the PRICE S model.

System-3 is another proprietary model. It is available for use with IBM-compatible personal computers, making it an

easily accessible model. System-3 accounts for software costs from product conception through the integration phase of software development (4:1-1). It is based on the Rayleigh-Norden curve and has some features that are also found in the SLIM and COCOMO models (7). System-3 has evolved over the years. The basic algorithms were developed by Dr Randall Jensen, but have been modified by Computer Economics, Inc. (CEI). System-3's predecessors were JS-1 and JS-2. The JS-1 was CEI's first estimating system, and "was the first in the industry to be both portable and free-standing" (4:1-3). It had only four primary input categories, one of which was a technology constant that evaluated the developer (4:1-3). This constant was determined by inputting 15 separate parameters. After a few years of validation, JS-2 evolved. It had many advantages over the JS-1. Development phase estimations were the same, but estimates for the requirements and the system integration phases were now possible. Also, "the user was able to fine-tune estimations with expanded technology and environment parameters" (4:1-4). This helped to improve the accuracy of estimates. Between JS-2 and System-3 came JS-DST, which was a DoD on-line decision support tool. It incorporated enhancements to JS-2 which were also included in System-3, the most current CEI software cost estimation tool.

System-3 considers many input factors: 7 developer factors, 17 environmental factors, size, complexity, and new design/code. As with the PRICE S variables, the System-3 variables will be discussed in much greater detail in Chapter III. The reader is referred to the System-3 manual (4), for a complete description of the System-3 model.

Related Studies

A 1977 study on software cost estimating methodology recognized the criticality of military software and the need for efforts to control software costs. James saw a need for a "defined method which will allow the analyst, software engineer, and/or project manager to estimate software development and support costs" (12:4). He discussed a number of methods for deriving software costs. Among them was a factors method which identified cost drivers and attempted to develop cost estimating relationships. James provided the results of a government/industry software workshop which identified eleven dominant cost drivers: (1) number of instructions in the program; (2) type of programming language; (3) real time application; (4) type of program; (5) desired quality; (6) amount of documentation; (7) hardware constraints; (8) schedules; (9) size of data bases; (10) complexity; and, (11) personnel and management functions (12:9-10). In 1977, there were a number of problems in the software cost estimation arena. There was a

need for "management control, visibility of the software structure, and a standardized framework for collecting historical costs, sizing, and requirement data" (12:11). Cost comparisons were difficult to make because there was "a lack of a common language, methodology, and work breakdown structure (WBS) which would provide a basis for developing and comparing cost estimates" (12:12). The Rome Air Development Center (RADC) proposed the creation of a centralized software data base which would provide the historical data needed to get good software cost estimates (12:16). In addition to discussing research going on then, James looked briefly at available models. He noted a major shortcoming in that none of the models had been adequately validated (12:24). He discussed the ongoing development of a PRICE software acquisition model by RCA. Also, he identified another shortcoming: "Yet to be developed are the quantitative effects on cost of using techniques such as structured programming, top-down development, chief programmer teams, and automated tools" (12:30). Today, PRICE S is a reality and many models consider the effects of the development techniques.

Schneider's 1977 AFIT thesis (20) looked at the PRICE S model when it was still very new and little cost data was available for model validation. In performing his study, Schneider considered characteristics of software systems, some of which could be objectively measured (ex. size), and

others which could only be measured subjectively (ex. complexity). Using historical data, he calibrated the PRICE S model and showed that "the PRICE S model is not incompatible with avionics software systems developed for the Aeronautical Systems Division of Air Force Systems Command" (20:ix). Since the PRICE S model has changed significantly since Schneider's study, no other results of his research will be presented.

A 1979 thesis by Steffey (22) also looked at the PRICE S model, but as it related to Air Force computer software acquisition and management. He looked at the model in depth and ran it 18 times. He used 18 different Air Force software development projects representing avionics, command and control, and management data systems (22:68). The resulting estimates showed that all of those systems could be adequately predicted within acceptable limits. Therefore, he concluded that the PRICE S model was indeed applicable to Air Force software cost estimating. Steffey's thesis also considerd the old PRICE S model, so specific results will not be discussed.

In 1981, Thibodeau (24) performed a detailed evaluation of nine software cost estimating models. His evaluation was typical of many others in describing qualitative aspects of the models. However, he also recognized that a standard of measurement should include how the models fulfill Air Force

software cost estimating needs. In his research, he made the following observations:

- The supporting materials for most of the models do not clearly state the elements included in their estimates and are not precise about their definitions.

- The existing models are better able to satisfy information needs early in the acquisition life cycle.

- None of the models included in this study fully satisfy the Air Force need for information either with regard to scope or detail.

- The models tend to be phase oriented and do not properly describe activities that cross phase boundaries. This precludes obtaining data compatible with both management planning (phase related) and product cost (WBS).

- Although most of the models use the summation of program or module sizes to make their cost estimates, only one model studied provides for keeping track of the cost *on a component* basis and accounts for the cost of system integration. None of the models provide for all four levels of system definition called for in the Work Breakdown Structure [24:5-1,5-2].

The model alluded to in the last observation was the PRICE S model. Many of the uncertainties involved with software cost estimating could be lessened if good data could be obtained. The necessary data would "include data that characterize the past performance in that environment as well as items of importance to the project being estimated" (24:7-1). The data issue remains a difficult problem even today.

With all the models available, it is reasonable to wonder if there is a "best model" to use. In his 1984

thesis, Steig (23) examined a procedure for selecting a model. He used discriminant analysis to see if models could be selected on the basis of the software type under development. He considered the PRICE S model and the Jensen model (a predecessor to System-3), in addition to two others. Steig hypothesized that:

> Through the use of pre-analysis determinants it will be possible to determine the software characteristics that correlate to the estimation accuracy of the tested model. Furthermore, this technique will reduce the total estimating error by selecting the estimating model most likely to provide the most accurate cost estimate [23:5].

Although Steig found that some software characteristics correlated to the accuracy of the models, he concluded that "a random selection of estimating models would have produced as much error as the selection of models using pre-analysis determinants" (23:39).

Yet another evaluation of some of the available software cost estimating models was conducted by Clark (3). He discussed six models in general terms and identified what he perceived as their benefits and weaknesses. Two of the models discussed were PRICE S and the Jensen model. Clark identified a key benefit of the PRICE S model in that it "is very good at providing software cost estimates during the early stages of program development when limited data is available" (3:10). A weakness Clark cited was the fact that PRICE S output was in terms of dollars rather than manmonths (PRICE S now allows both). Clark saw the Jensen model as

being user friendly, with very good documentation (3:24).

The Jensen model weakness that he identified was as follows:

> The Jensen model produces irregular relationships between person-months of effort and lines of code produced when either the software development projects are longer than five years or when software development teams comprise less than five people [3:24].

Clark leaves the selection of the model to use to the reader, but offers the suggestion of using evaluation criteria laid out by Boehm (2:476) as a decision tool (3:27-28).

This thesis will look at the relationship between the variables used by the PRICE S model and the System-3 model. No literature was found that covered such research. Therefore, the study of the relationship between the variables used by these two models could be a fruitful endeavor.

III. Methodology

This chapter is divided into three main sections. The first section, "Examination of Inputs," involves an examination of the input variables used by the PRICE S and System-3 models to determine the software characteristics used by the models to estimate costs. The second section, "Comparability of Inputs," evaluates the comparability of the variables used by the models to determine the relationships needed for comparison purposes. The third section, "Sensitivity Analysis," describes activities that are crucial to defining the domain of consistency for each of four software development efforts. The sub-sections under the "Sensitivity Analysis" section include "PRICE S Software Project Constructs," "PRICE S Estimates," "System-3 Estimates" and, "Domain of Consistency."

Examination of Inputs

The first step is to examine the inputs used by the individual models. The information in this section comes from the PRICE S Reference Manual (19: Section III) and the System-3 Reference Manual (4: Chapter 7). A more detailed description of the variables used by the PRICE S model and the System-3 model can be found in Appendices A and B, respectively.

The PRICE S model has relatively few input variables. PRICE S has eleven basic inputs: (1) total number of source

lines of code to be developed (SLOC), (2) inherent instruction complexity (APPL), (3) a calibration of the net effects of such organizational tendencies as skill levels, experience, productivity, and efficiency (PROFAC), (4) the fraction of available hardware cycle time or total memory capacity used (UTIL), (5) the customer's requirements stemming from his planned operating environment (PLTFM), (6) a quantitative description of the relative effect of complicating factors on the software task (CPLXM), (7) a quantitative description of the relative effect of complicating factors on the software development task (CPLX1), (8) a quantitative description of the relative effect of complicating factors on the software development task caused by hardware/software interactions (CPLX2), (9) amount of new design (NEW DESIGN), (10) amount of new code (NEW CODE), and (11) the language the software is written in (LANG) (19). For ease of reference, Table 2 lists the PRICE S acronymns and the variables they represent.

The System-3 model has 24 technology and environment input variables, as well as a few other separate inputs. The technology and environment variables are divided into four different categories. Application complexity (APPL), analyst capability (ACAP), analyst applications experience (AEXP), the use of modern design practices (MODP), programmer capability (PCAP), automated tool support (TOOL), and turnaround time (TURN) comprise the

Table 2

PRICE S Variables

Acronym	Variable
APPL	User Defined Application (Inherent Instruction Complexity)
CPLXM	Effect of Complicating Factors On Software Task
CPLX1	Effect of Complicating Factors on Software Development Task
CPLX2	Effect of Complicating Factors on Software Development Task Caused by Hardware/Software Interactions
LANG	Language
NEW CODE	Amount of New Code
NEW DESIGN	Amount of New Design
PLTFM	Customer's Requirements Stemming From Planned Operating Environment
PROFAC	Productivity Factor
SLOC	Source Lines of Code
UTIL	Fraction of Available Cycle Time or Total Memory Capacity Used

developer technology category. The environment technology in terms of computer related environment parameters consists of special display requirements (DISP), memory constraints (MEMC), time constraints (TIMC), and real time operation (RTIM). The environment technology in terms of product related environment parameters consists of specification level (SPEC), quality assurance level (QUAL), test level (TEST), requirements change - volatility (RCHG),

rehosting development (HOST), language type rating (LANG), language experience (LEXP), system (virtual machine) complexity (SYST), system (virtual machine) experience (SEXP), and virtual machine volatility (VMVL). Finally, the three parameters that comprise the environment technology in terms of support related environment parameters are multiple site development (MULT), resource dedication (RDED), and resource/support location (RLOC). There are four additional variables in the System-3 model that need to be addressed. Lines of code (LOC) represents size and accounts for non-comment source lines of code. Complexity is a factor that specifies the relative complexity of the specific software component. System-3 also allows specification of the amount of new design and new code. There are no acronyms for these last three System-3 variables. For ease of reference, Table 3 lists the System-3 acronymns and the variables they represent.

Table 3

System-3 Acronyms (adapted from 4:i2-i3)

Acronym	Variable
ACAP	Analyst CAPability
AEXP	Application EXPerience
APPL	APPLication Class Complexity
DISP	Special DISPlay Requirements
HOST	ReHOSTing

Table 3 (cont.)

Acronym	Variable
LANG	LANGuage Type Rating
LEXP	Language EXPerience
LOC	Lines Of Code
MEMC	MEMory Constraints
MODP	MOdern Development Practices
MULT	MULTiple Site Development
PCAP	Programmer CAPability
QUAL	QUALity Assurance Level
RCHG	Requirements CHanGe - Volatility
RDED	Resource DEDication
RLOC	Resource LOCation
RTIM	Real TIMe Operation
SEXP	System (Virtual Machine) EXPerience
SPEC	SPECification Level
SYST	SYSTem (Virtual Machine) Complexity
TEST	TEST Level
TIMC	TIMe Constraint
TOOL	TOOL Support, Automated
TURN	TURNaround - Logon Through Hardcopy
VMVL	Virtual Machine VoLatility

Comparability of Inputs

The difference in the number of input variables used by the two models would seem to imply that there is not strict

commonality between the models. Upon further examination, that is clearly the case. However, there are some similarities as well as some differences. The System-3 model divides the characteristics of the software task and the software development environment into very specific areas, while the PRICE S model includes many characteristics within one input variable. This makes the task of correlating the variables of one model with those of the other quite difficult, and a large amount of subjectivity is needed to make some comparisons. In fact, Dr Robert Park of PRICE Systems, Dr Barry Boehm (COCOMO model), and Dr Lawrence Putnam (SLIM model) attempted to objectively compare the input variables used by a number of software cost estimating models, but were unable to do so (16). Since it cannot be done objectively, some subjectivity is required when comparing the input variables. This may detract from the ability to generalize the findings of this study.

Since the sensitivity analysis will involve running the PRICE S model first (as will be discussed in the "Sensitivity Analysis" section of this chapter), the PRICE S variables will be discussed in terms of their possible relationships with System-3 variables.

There appears to be a one-to-one correspondence between the PRICE S LANG variable and the System-3 LANG variable. In PRICE S, LANG is the name of the source language to be

used for the software development effort. In System-3, LANG represents the language type rating, and is the primary language to be used during development. Internally, PRICE S handles LANG as a technical improvement so it may have a different effect on cost than the LANG variable in System-3. There also seems to be a direct comparison between NEW DESIGN and NEW CODE in PRICE S with new design and new code in System-3.

CPLXM in PRICE S represents management complexity and takes into account multinational projects and/or software developed at more than one location. This variable relates to the System-3 MULT variable, which represents multiple site development and rates the inefficiency due to site and organizational diversity.

At first glance, it appears that the PRICE S variable, SLOC, relates to the System-3 variable LOC. However, SLOC represents the total number of source lines of code to be developed, including type declarations and data statements; but, PRICE S accounts for type declarations and data statements with the sub-variable FRAC, which represents the fraction of non-executable code. Such handling of lines of code allows executable lines of code to be priced at one level, and non-executable lines of code to be priced at another. On the other hand, LOC includes data declaration statements as well as all executable source lines, but there is no difference implied in cost between executable and non-

executable lines of code. In essence, it appears that SL and LOC are nearly the same, but PRICE S uses FRAC to differentiate between executable and non-executable lines code. The different definitions for these variables must accounted for when doing the sensitivity analysis, becaus possible cost differences between executable and non-executable lines of code represent a possible weakness in the comparison of the models. The remaining variables ar more difficult to compare.

The APPL variable in the PRICE S model does not directly correspond to the APPL variable in the System-3 model. APPL in PRICE S summarizes the application mix of instructions, while APPL in System-3 represents the application class complexity rating which establishes the overall level of system application. However, APPL in PRICE S gives an inherent instruction complexity. This makes the PRICE S APPL variable more compatible with the System-3 complexity variable, which specifies the softwar component's difficulty independent of the developer's ability to implement the component. By looking at the possible categories for the PRICE S APPL variable and for complexity in System-3, the correspondence between the tw variables is easier to see. Table 4 shows the instructio mix that makes up APPL in the PRICE S model, while Table gives values for complexity in System-3. APPL in PRICE S can be computed internally if mix percentages are specifi

Table 4

PRICE S APPL Mix Elements (Adapted From 19:II-A13)

Application Weight	Instruction Mix
.86	Mathematical Operations
2.31	String Manipulation
4.10	Data Storage and Retrieval
6.16	On-Line Communications
8.46	Real Time Command and Control
10.95	Interactive Operations

Table 5

System-3 Complexity Factors (adapted from 4:7-8,7-9)

Rating	Value	Software Component's Difficulty
Extra High	4	Development primarily using microcode for the application
Very High	8	New systems with significant interface and instruction requirements
High	11	Application program with significant logical complexity
Nominal	15	New standalone systems developed on firm operating systems
Low	21	Software of low logical complexity
Very Low	28	Extremely simple software

or input directly. For example, if the nature of the software was such that 50% was string manipulation and 50% was on-line communication, the APPL value would be computed as follows:

```
.5 x 2.31   =   1.155
.5 x 4.10   =   2.050
                -----
APPL value  =   3.205
```

Following similar computations, the APPL values of typical systems were computed. A management information system could have an APPL of 2.09, satellite software an APPL of 3.77, fire control systems an APPL of 5.42, mobile radar an APPL of 7.13, and for an operating system, an APPL of 10.95 (18:III-5).

An observation from Tables 4 and 5 is that in PRICE S, the APPL value increases as the application becomes more complex, while the complexity rating in System-3 decreases as the software component increases in difficulty. For System-3, mil-std software is no more complex than 7, and no less simple than 15.

Since APPL in System-3 represents the overall level of system application, it relates more to the PRICE S variable, PLTFM, which describes the customer's requirements stemming from his planned operating environment. Table 6 lists typical PLTFM values, while Table 7 gives typical ratings for the System-3 APPL variable.

Table 6

Typical PLTFM Values (Adapted From 19:II-A2)

PLTFM	Operating Environment
0.6 - 0.8	Production Center Internally Developed S/W
1.0	Production Center Contracted S/W
1.2	MIL-Spec Ground
1.4	Military Mobile (Van or Shipboard)
1.7	Commercial Avionics
1.8	MIL-Spec Avionics
2.0	Unmanned Space
2.5	Manned Space

Table 7

System-3 APPL Values (4:7-7)

Rating	Value	Type of Applications
Low	1.0	Business Data Processing Applications
	1.3	Signal Processing
	1.5	Interface Systems
Nominal	2.0	Applications With Complex Systems, File and/or User Interfaces, Such as Command and Control
	2.0	Communications Networks
	2.5	Command and Control Systems
High	3.0	Networks, Operating Systems, Compilers
	3.0	Fire Control Systems

To a lesser degree, PLTFM relates to four other System-3 variables in addition to APPL. PLTFM "is a measure of the portability, reliablity, structuring, testing, and documentation required for acceptable contract performance" (19:II-A2). Therefore, PLTFM incorporates the concepts included in the System-3 variables. DISP, SPEC, QUAL, and TEST. Special display requirements (DISP) rates the extra effort required to respond to changing user needs. Specification level (SPEC) rates the required design, code, test, and other documentation level. Quality assurance level (QUAL) is the depth with which quality assurance will be used for the software, and test level (TEST) is the depth at which the software will undergo testing by the developer organization.

UTIL (PRICE S) seems to correspond to three variables in System-3. Since UTIL describes the extra effort needed to adapt software to operate within limited processor capabilities, it closely relates to MEMC, RTIM, AND TIMC, which break that effort into three areas. MEMC represents memory constraints that measure the probable amount of extra effort required to fit the software into the target machine; RTIM is real time operation that rates the amount of source code that must handle externally timed operations; and, TIMC represents time constraints code that must have special attention to ensure adequate time performance (4:7-22, 7-34,7-57).

The PRICE S variable, PROFAC, is a productivity factor which reflects the overall efficiencies of an organization. It is empirically derived (from historical data) and includes skill levels, experience, productivity, and efficiency (19:II-A10). PROFAC provides a means to calibrate the model to the typical productivity achieved by an organization. If the organization's typical productivity is known, a value for PROFAC may be input directly. The corresponding System-3 variables are resource dedication (RDED), and resource and support location (RLOC). RDED rates the availability of development computers and other computing resources, while RLOC rates the time wasted by developers in accessing support personnel and tools. The PRICE S model assumes that development computers and other computing resources will be fully dedicated throughout the development cycle.

The CPLX1 complexity variable in the PRICE S model encompasses a number of System-3 variables. "Factors entering into CPLX1 are product familiarity, personnel skills, software tools, and any unusual factors present in the development environment that affect the development schedule" (19:II-A8). Therefore, the corresponding System-3 variables include ACAP, AEXP, PCAP, LEXP, MODP, TOOL, HOST, SYST, and SEXP. Analyst capability (ACAP) measures the capability of the analyst software team assigned to a specific project over the duration of the project, and

analyst experience (AEXP) is the project team's equivalent experience level with this general type of application. Programmer capabilities (PCAP) rates the ability of the programming team to function effectively, and language experience (LEXP) rates the team's experience with similar programming languages at the start of full scale development. Modern development practices (MODP) rates the developer's use of modern software development methods and practices. Automated tool support (TOOL), rehosting (HOST), system complexity (SYST), and system experience (SEXP) all relate to tooling. TOOL rates the support of development practices with automated software development tools; HOST represents movement from original development facilities to other machines, compilers, programming languages, etc.; SYST rates the inherent difficulty in learning and richness of the development computing resources; and, SEXP rates the level of experience the developers have with the virtual machine at the start of full scale development.

CPLX2 is a PRICE S variable which includes factors such as new hardware development and hardware developed in parallel. The PRICE S model assumes that development computing resources will remain constant throughout the development cycle. CPLX2 does not correspond to any System-3 variable.

The System-3 variables VMVL and TURN do not have PRICE S equivalents. Virtual machine volatility (VMVL)

rates the rate of changes within the development computing resources, and turnaround time (TURN) is the number of hours wasted time from logging onto the development system until hardcopy output is obtained. Table 8 summarizes the relationships between the variables used by the PRICE S model and the System-3 model.

Sensitivity Analysis

There are two different approaches to conducting a traditional sensitivity analysis on a model. One approach is to test the sensitivity of the model output to different values for the inputs. The model parameters are held constant and the inputs are allowed to vary:

> the analyst may successively use several values (say, high, medium, and low) in an attempt to see how sensitive the results (the ranking of the alternatives being considered) are to variations in the uncertain parameters [i.e., inputs] [11:12].

The second approach to sensitivity analysis is to evaluate the sensitivity of the model output to changes in the model parameters. The inputs are held constant and the model parameters are allowed to vary. Frank offers a reason for such an approach:

> it should be part of the solution to a practical problem to know the parameter sensitivity prior to its implementation or to reduce the sensitivity systematically if this turns out to be necesary [12:1].

These two approaches may produce very different results. The developers of the PRICE S and System-3 models have already performed sensitivity analyses on their models,

Table 8

Summary of Relationships

	PRICE S	System-3
One-to-One	LANG	LANG
	NEW CODE	New Code
	NEW DESIGN	New Design
	APPL	Complexity
	CPLXM	MULT
	SLOC (when FRAC=0)	LOC
One-to-Many	UTIL	MEMC RTIM TIMC
	PLTFM	APPL DISP QUAL SPEC TEST
	PROFAC	RDED RLOC
	CPLX1	ACAP AEXP HOST LEXP MODP PCAP RCHG SEXP SYST TOOL
No Match	CPLX2	
		TURN
		VMVL

using the first approach. Since the models are proprietary, the parameters are unknown and sensitivity analyses using the second approach cannot be performed.

This study takes a different approach by identifying a domain of values for selected variables from one model (System-3) that would generate estimates reasonably close (+/-5%) to an estimate produced by a second model (PRICE S), for a given software development effort. The analysis demonstrates under what conditions it would be reasonable to expect the two models to produce similar estimates and, consequently, under what conditions the two models would produce very different estimates.

PRICE S Software Project Constructs. The PRICE S model will be run with sets of input values representing four software development efforts: simple, moderate, complex, and very complex. These are hypothetical programs based on possible values for the PRICE S inputs. All four efforts describe MIL-spec avionics developed by aerospace organizations.

The simple effort represents a relatively small Fortran program primarily dealing with data storage and retrieval, and requiring little new code or design. Also, it requires a normal fraction of the available hardware cycle time or total memory capacity. It is developed by a normal crew for this type of effort, that is, less experienced personnel can be used. Lastly, it is developed at a single location.

The moderate effort describes a Pascal program primarily dealing with on-line communications. It requires slightly more new code/design and a higher fraction of hardware cycle time/total memory capacity than the simple effort. It is developed by personnel with mixed experience, at a single location.

The complex effort represents a Jovial program primarily dealing with real time command and control. It requires slightly more new code/design and a higher fraction of hardware cycle time/total memory capacity than the moderate effort. Requirements for the software are expected to change somewhat. The effort will be performed by experienced crews at more than one location.

Finally, the very complex effort describes an Ada program primarily dealing with interactive operations or operating systems. It requires slightly more new code/design and a higher fraction of hardware cycle time/total memory capacity than the complex effort. There are expected to be many changing requirements, and Ada can be considered a relatively new language. This effort will be carried out by experienced personnel, as a multinational project.

PRICE S Estimates. Four inputs into the PRICE S model were held constant for all four software development efforts. The PLTFM and PROFAC variables were set at values that closely represent work done at ASD. PLTFM was set at

1.8 to indicate mil-spec avionics, and PROFAC was set at 4.0 to indicate the efficiency of a typical ASD contractor. The settings for PLTFM and and PROFAC were based on conversations with James Otte of PRICE Systems (16). Since CPLX2 does not have a System-3 equivalent, it was normalized at 1.0 (the general defense industry average). Additionally, FRAC was set to 0.0, so that there can be more of a relationship between SLOC and the System-3 variable, LOC.

The remaining inputs were set at values that best corresponded to the level of complexity of each of the four efforts. The estimates that resulted from the four PRICE S runs will serve as a baseline for the System-3 model's estimates. A variance of plus or minus five percent will be allowed for those estimates, thus giving an interval for the System-3 estimates.

System-3 Estimates. Since PROFAC was held constant, its System-3 equivalents were also held constant. RDED was set to 10.0, which indicates that there is 100% access to computing resources (i.e. fully dedicated). This is compatible with the Price-S model which assumes that development computers and other computing resources will be fully dedicated throughout the development cycle (16). RLOC is set to 0.0 (a nominal rating for System-3), indicating that developers waste little time in accessing support personnel and tools.

The System-3 equivalents to PLTFM must also be held constant, by setting them to values that represent mil-spec avionics. APPL was set to 3.0, indicating computer networks, operating systems, compilers, fire control systems, or complex signal acquisition and processing. The setting for APPL was based on conversations with Wayne Stanley of CEI (23). SPEC was set to 6.0 to indicate mil-spec full documentation; QUAL was set to 4.0, indicating mil-spec quality assurance; and, TEST was set to 4.0, indicating normal testing by the developer.

Since the System-3 variables TURN and VMVL do not have PRICE S equivalents, they were assigned values that would minimize their impacts on the estimate. TURN was set to 2.0 (a nominal rating), indicating that it takes two hours from the time of logging onto the development system until hardcopy output is obtained. VMVL was set to 0.0, indicating no major changes within the development computing resources and only minor changes each year. This is compatible to the Price-S model, which assumes that development computing resources will remain constant throughout the development cycle.

Six System-3 inputs that were assigned different values for each software project were not subjected to a sensitivity analysis. These six inputs were LANG, New Code, New Design, Complexity, MULT, and LOC. All six inputs have a one-to-one correspondence to PRICE S inputs. Therefore, a

change in the values of these inputs would alter the basic definition of the software project being estimated. This was not the case where groups of System-3 inputs corresponded to single PRICE S inputs. There were ten System-3 variables that corresponded to the PRICE S variable CPLX1, and three System-3 variables that corresponded to the PRICE S variable UTIL (refer to Table 8). These System-3 inputs could be allowed to vary and still be compatible with the single PRICE S value. The CPLX1 and UTIL related variables were considered separately.

Those System-3 inputs that were allowed to vary were evaluated from three different approaches. First, each individual input was allowed to vary with the other inputs held constant. This analysis would define the range of values that an individual input could assume and still have the model produce an estimate reasonably close to the PRICE S estimate.

Although System-3 has designated ratings for its input variables, it allows the variables to change on a continuum. Therefore, there are an infinite number of possible entries for a single variable. Consequently, the number of possible combinations that can result in estimates that are within the interval around the PRICE S estimate is also infinite. As a result, changes in variables as a group will be considered in terms of percentage changes only.

The second approach involves changing the variables in such a way that they all drive cost in the same direction. This means that for a cost increase, inputs that are positively correlated with cost would increase, and inputs that are negatively correlated with cost would decrease. Conversely, for a cost decrease, inputs that are positively correlated with cost would decrease, and inputs that are negatively correlated with cost would increase. This approach will be used for both the CPLX1 and UTIL related variables.

The third approach involves increasing or decreasing all of the inputs in the group at the same time. This approach allows for trade-offs between the effects of different inputs on cost when some inputs are positively correlated with cost and some are negatively correlated with cost. The System-3 inputs that relate to CPLX1 in PRICE S meet this criteria. RCHG, HOST, and SYST are positively correlated with cost, while the other seven inputs are negatively correlated with cost. By increasing RCHG, HOST, and SYST, the other CPLX1 related variables can be increased as well (and vice versa), while keeping the resulting estimate within the interval about the PRICE S estimate. These trade-offs indicate that less difficult work is assigned to less able teams, or that more difficult work is assigned to more able teams. This approach will only be

used for the CPLX1 related variables since all of the UTIL related variables are positively correlated with cost.

Domain of Consistency. The results of the sensitivity analysis will determine the domain of consistency for each of the four software development efforts. That is, the ranges of System-3 input values for which the resulting estimates approximate the PRICE S estimates will be identified. By identifying the values for the input variables for which the models give similar estimates, it can be seen that either model can be used for estimating purposes. The main point to note here is that the variables entered into the models must represent similar software/software development environments for the domain of consistency to be valid. These findings are found in Chapter IV.

IV. Analysis and Findings

This chapter includes a presentation of specific values used for the four runs of the PRICE S model, as well as the specific values used for the initial runs of the System-3 model. It also discusses the results of the sensitivity analyses on the System-3 variables that related to the PRICE S variables CPLX1 and UTIL. All four software development efforts will be discussed.

PRICE S Model

The PRICE S model was run first, to provide a baseline for the System-3 model runs.

Values for PRICE S Inputs. As discussed in Chapter III, the values for PROFAC, PLTFM, and CPLX2 are held constant across all four software projects. PLTFM was set at 1.8, PROFAC was set at 4.0, and CPLX2 was set at 1.0. Table 9 provides reasonable values for the other PRICE S input variables to represent the four software efforts of increasingly complex levels of performance. For purposes of this study, these defined inputs will change from project to project. The reader may refer to Appendix A for further explanation of these variables. Outputs for the PRICE S runs for the simple, moderate, complex, and very complex software project constructs can be found in Appendices C, D, E, and F, respectively. These outputs include a complete listing of the PRICE S input variables.

Table 9

PRICE S Input Values

Variable	Simple	Moderate	Complex	Very Complex
APPL	4.1	6.16	8.46	10.95
CPLXM	1.0	1.00	1.20	1.40
CPLX1	0.9	1.10	1.20	1.30
LANG	Fortran	Pascal	Jovial	Ada
NEW CODE	0.4	0.50	0.70	0.80
NEW DESIGN	0.4	0.50	0.70	0.80
SLOC	10000	25000	40000	60000
UTIL	0.5	0.60	0.70	0.80
ESTIMATE	55	288	1055	4709

Purpose of PRICE S Estimates. The resulting estimates from the PRICE S runs for each of the four software project constructs will serve as a baseline for the sensitivity analysis on the System-3 inputs. That is, the System-3 variables were adjusted so that the resulting System-3 estimate would be within plus or minus five percent of the PRICE S estimate. A variance of plus or minus five percent was selected as the standard for defining an appropriate interval because 5% is a reasonable estimating error. The PRICE S estimates and their related bounds are listed in Table 10.

Table 10

PRICE S Estimates and Bounds

Software Project	Lower	Estimate	Upper
Simple	52.3	55	57.8
Moderate	273.6	288	302.4
Complex	1002.3	1055	1107.8
Very Complex	4473.6	4709	4944.5

System-3 Model

The System-3 model was run a number of times in the attempt to find the domains of consistency.

Values for System-3 Inputs. As discussed in Chapter III, the System-3 variables corresponding to PLTFM and PROFAC, as well as those variables unique to the System-3 model, are held constant. APPL was set at 3.0, DISP was set at 6.8, QUAL was set at 4.0, SPEC was set at 6.0, TEST was set at 4.0, RDED was set at 10.0, RLOC was set at 0.0, TURN was set at 2.0, and VMVL was set at 0.0, for all four constructs. For each of the four software development projects, the remaining System-3 inputs were assigned values that seemed to best correspond to their related PRICE S variables. Table 11 provides the assigned values for the remaining System-3 inputs. All of these variables can change from project to project, but only the CPLX1 related variables and the UTIL related variables will be subjected to a sensitivity analysis. The reader may refer

Table 11

System-3 Input Values

	Variable	Simple	Moderate	Complex	Very Complex
	LOC	10000	25000	40000	60000
	New Code	0.4	0.5	0.7	0.9
	New Design	0.4	0.5	0.7	0.9
	Complexity	15	11	8	7
*	ACAP	3.5	3.5	5.5	5.5
*	AEXP	1.0	1.0	1.0	1.0
*	MODP	3.0	3.0	3.0	5.2
*	PCAP	3.5	3.5	5.5	5.5
*	TOOL	3.0	5.2	5.2	7.5
+	MEMC	1.0	2.0	5.5	6.0
+	TIMC	5.0	6.0	7.0	8.0
+	RTIM	5.0	6.0	7.0	8.0
*	RCHG	1.3	4.0	7.0	10.0
*	HOST	1.0	2.9	5.1	6.9
	LANG	1.0	1.0	2.0	3.0
*	LEXP	1.0	1.0	1.0	0.5
*	SYST	2.0	2.0	2.0	3.0
*	SEXP	1.0	1.0	1.0	0.5
	MULT	0.0	0.0	6.5	10.0

* CPLX1 related variables

\+ UTIL related variables

to Appendix B for further explanation of these variables. Outputs for the System-3 runs for the simple, moderate, complex, and very complex software development projects can be found in Appendices C, D, E, and F, respectively. These outputs include a complete listing of the System-3 inputs.

Simple Project. Individual changes in the "CPLX1 variables" were looked at first, and the results are listed in Table 12. One of the reasons that the variables can change more to arrive at the lower bound than they can to arrive at the upper bound is that the initial System-3 estimate was slightly higher than the PRICE S estimate. In terms of designated System-3 ratings for the variables (refer to Appendix B), most of the individual changes are not very large. This is not the case for AEXP, MODP, TOOL, LEXP, and SYST.

ACAP can increase to 4.3 without the estimate falling below the lower bound, but the next highest specific rating is 5.5. Therefore, this new value is still closer to its original value of 3.5. The same holds true for PCAP. Also, RCHG, HOST, and SEXP are allowed only minor changes, indicating that they have a relatively strong impact on cost. In other words, the model is more sensitive to changes in these variables.

It should be noted that MODP and TOOL have the same effect on cost. The changes in MODP and TOOL represent significant changes, since they can increase to 4.8, which

Table 12

Ranges for Changes in Individual "CPLX1" Variables
(Simple Project)

Variable	Baseline Value [Est: 55.9]	Low Value	Low [Est]	High Value	High [Est]
# ACAP	3.5	4.3	[52.3]	3.2	[57.3]
# AEXP	1.0	1.8	[52.6]	0.8	[57.2]
# MODP	3.0	4.8	[52.3]	2.2	[57.6]
# PCAP	3.5	4.3	[52.5]	3.1	[57.7]
# TOOL	3.0	4.8	[52.3]	2.2	[57.6]
+ RCHG	1.3	0.3	[52.4]	1.8	[57.7]
+ HOST	1.0	0.4	[52.5]	1.3	[57.6]
# LEXP	1.0	* 1.1	[55.2]	0.6	[57.4]
+ SYST	2.0	1.6	[52.8]	2.2	[57.6]
# SEXP	1.0	1.3	[52.9]	0.9	[57.2]

Negatively correlated -- estimates decrease as variables increase

+ Positively correlated -- estimates increase as variables increase

* Values greater than 1.1 result in an estimate of 55.2

is very close to the next highest specific rating, 5.2. That means that larger changes are needed to have an impact on cost. In other words, the model is less sensitive to changes in these variables. The same holds true for AEXP, LEXP, and SYST.

Changes in the variables as a group were looked at next. Table 13 shows the changes that drive cost in the

Table 13

"CPLX1" Changes as a Group: Drive Cost in Same Direction (Simple Project)

	Variable	-2.0%	-1.5%	Baseline	+3.5%	+4.0%
	ACAP	3.43	3.45	3.5	3.62	3.64
	AEXP	0.98	0.98	1.0	1.04	1.04
	MODP	2.94	2.96	3.0	3.11	3.12
	PCAP	3.43	3.45	3.5	3.62	3.64
	TOOL	2.94	2.96	3.0	3.11	3.12
*	RCHG	1.33	1.32	1.3	1.25	1.25
*	HOST	1.02	1.02	1.0	1.04	1.04
	LEXP	0.98	0.98	1.0	1.04	1.04
*	SYST	2.04	2.03	2.0	1.93	1.92
	SEXP	0.98	0.98	1.0	1.04	1.04
	ESTIMATE	57.80	57.30	55.9	52.30	51.90

* Changes take opposite signs

same direction, while Table 14 shows the changes that involve trade-offs. Table 13 indicates that the variables can change by 3.5% (+3.5% for negatively correlated variables and -3.5% for positively correlated variables) without falling below the lower bound; and, can change by 2% (-2% for negatively correlated variables and +2% for positively correlated variables) without exceeding the upper bound. Table 14 shows that the variables can increase by

Table 14

"CPLX1" Changes as a Group: Trade-offs
(Simple Project)

Variable	-6%	-5%	Baseline	+7%	+8%
ACAP	3.29	3.33	3.5	3.75	3.78
AEXP	0.94	0.95	1.0	1.07	1.08
MODP	2.82	2.85	3.0	3.21	3.24
PCAP	3.29	3.33	3.5	3.75	3.78
TOOL	2.82	2.85	3.0	3.21	3.24
RCHG	1.22	1.24	1.3	1.39	1.40
HOST	0.94	0.95	1.0	1.07	1.08
LEXP	0.94	0.95	1.0	1.07	1.08
SYST	1.88	1.90	2.0	2.14	2.16
SEXP	0.94	0.95	1.0	1.07	1.08
ESTIMATE	58.10	57.70	55.9	52.9	52.6

approximately 8% without falling below the lower bound, and can decrease by approximately 5% without exceeding the upper bound, when all variables are changed in the same direction.

The domain of consistency is represented by any changes in the variables that do not exceed the limits specified above. On the other hand, the domain of inconsistency is obtained when those limits are exceeded.

The same procedure was followed for the UTIL related variables. Table 15 gives the ranges for changes in individual variables, while Table 16 gives the results of

Table 15

Ranges for Changes in Individual "UTIL" Variables
(Simple Project)

Variable	Baseline Value [Est: 55.9]	Low Value	Low [Est]	High Value	High [Est]
MEMC	1.0	0.0	[53.3]	1.7	[57.7]
TIMC	5.0	3.8	[52.4]	5.6	[57.7]
RTIM	5.0	3.4	[52.3]	5.8	[57.7]

** All of the "UTIL" variables are positively correlated with cost

changes in the variables as a group that drive cost in the same direction. The results in Table 15 indicate that small changes in MEMC, TIMC, and RTIM drive the estimate towards the bounds. That is, the model is sensitive to changes in these variables. RTIM has the least impact on cost (i.e., of the three, the model is least sensitive to RTIM) and therefore can change the most, before surpassing the bounds. TIMC has the greatest impact on cost and therefore can change the least. Although it may appear that MEMC changes the least (1.0 to 0.0), it must be noted that a value of 0.0 results in an estimate of 53.3, which is greater than the lower bound by 1.0. Table 16 indicates that the "UTIL" variables can decrease by approximately 12% without falling below the lower bound, while increases in the "UTIL" variables are limited to 7%. The domain of consistency is achieved as long as changes in the variables do not exceed

Table 16

"UTIL" Changes as a Group: Drive Cost in Same Direction (Simple Project)

Variable	-12%	-11%	Baseline	+6%	+7%
MEMC	0.88	0.89	1.0	1.06	1.07
TIMC	4.40	4.45	5.0	5.30	5.35
RTIM	4.40	4.45	5.0	5.30	5.35
ESTIMATE	52.50	52.80	55.9	57.60	57.90

the limits specified above. If the changes do exceed those limits, the variables are then considered to be in the domain of inconsistency.

Moderate Project. Individual changes in the "CPLX1 variables" were looked at first, and the results are listed in Table 17. Many of the variables may not seem to approach their bounds because the next incremental change (of 0.1) caused the estimates to surpass those bounds. For example, a SEXP value of 1.3 resulted in an estimate that was less than the lower bound of 273.6, while a SEXP value of 0.8 resulted in an estimate greater than the upper bound of 302.4. The same relative impacts on cost as discussed for the simple project apply here. In relative terms, ACAP, PCAP, RCHG, HOST, and SEXP have smaller changes than AEXP, MODP, TOOL, LEXP, and SYST, because they have greater impacts on cost (the model is more sensitive to them).

Table 17

Ranges for Changes in Individual "CPLX1" Variables
(Moderate Project)

Variable	Baseline Value [Est: 288.5]	Low Value [Est]	High Value [Est]
# ACAP	3.5	4.1 [274.4]	3.0 [300.8]
# AEXP	1.0	1.6 [274.7]	0.7 [298.9]
# MODP	3.0	4.4 [274.0]	1.8 [301.5]
# PCAP	3.5	4.1 [275.4]	2.9 [302.2]
# TOOL	5.2	6.6 [274.0]	4.0 [301.5]
+ RCHG	4.0	3.1 [274.1]	4.8 [301.4]
+ HOST	2.9	2.4 [275.7]	3.4 [301.3]
# LEXP	1.0	* 1.1 [285.2]	0.5 [300.1]
+ SYST	2.0	1.7 [276.1]	2.3 [302.0]
# SEXP	1.0	1.2 [277.7]	0.9 [295.1]

Negatively correlated -- estimates decrease as variables increase

+ Positively correlated -- estimates increase as variables increase

* Values greater than 1.1 result in an estimate of 285.2

Changes in the "CPLX1" variables as a group were looked at next. Table 18 shows the changes that drive cost in the same direction, while Table 19 shows the changes that involve trade-offs. Table 18 indicates that the variables can change by 2.5% in both directions (for decreases: +2.5% for negatively correlated variables and -2.5% for positively correlated variables, and vice versa) without exceeding

Table 18

"CPLX1" Changes as a Group: Drive Cost in Same Direction (Moderate Project)

	Variable	-2.5%	-2.0%	Baseline	+2.5%	+3.0%
	ACAP	3.41	3.43	3.5	3.59	3.61
	AEXP	0.98	0.98	1.0	1.03	1.03
	MODP	2.93	2.94	3.0	3.08	3.09
	PCAP	3.41	3.43	3.5	3.59	3.61
	TOOL	5.07	5.10	5.2	5.33	5.36
*	RCHG	4.10	4.08	4.0	3.90	3.88
*	HOST	2.97	2.96	2.9	2.83	2.81
	LEXP	0.98	0.98	1.0	1.03	1.03
*	SYST	2.04	2.03	2.0	1.93	1.92
	SEXP	0.98	0.98	1.0	1.04	1.04
	ESTIMATE	302.80	300.30	288.5	273.60	271.80

* Changes take opposite signs

bounds. Table 19 shows that the variables can increase by approximately 10% without falling below the lower bound, and decrease by approximately 11% without exceeding the upper bound, when all variables are changed in the same direction.

The same procedure was followed for the UTIL related variables. Table 20 gives the ranges for changes in individual variables, while Table 21 gives the results of changes in the variables as a group that drive cost in the

Table 19

"CPLX1" Changes as a Group: Trade-offs
(Moderate Project)

Variable	-11%	-10%	Baseline	+10%	+11%
ACAP	3.12	3.15	3.5	3.85	3.89
AEXP	0.89	0.90	1.0	1.10	1.11
MODP	2.67	2.70	3.0	3.30	3.33
PCAP	3.12	3.15	3.5	3.85	3.89
TOOL	4.63	4.68	5.2	5.72	5.77
RCHG	3.56	3.60	4.0	4.40	4.44
HOST	2.58	2.61	2.9	3.19	3.22
LEXP	0.89	0.90	1.0	1.10	1.11
SYST	1.78	1.80	2.0	2.20	2.22
SEXP	0.89	0.90	1.0	1.10	1.11
ESTIMATE	302.50	301.50	288.5	273.90	272.70

same direction. Table 20 indicates that once again TIMC has the smallest change, followed by MEMC and then RTIM. Table 21 indicates that the "UTIL" variables can decrease by approximately 8.5% and stay above the lower bound, and can increase by approximately 7.5% without exceeding the upper bound. Once again, the domain of consistency is defined by changes in the variables that do not exceed the percentages specified above. If those percentages are exceeded, the variables represent the domain of inconsistency.

Table 20

Ranges for Changes in Individual "UTIL" Variables
(Moderate Project)

Variable	Baseline Value [Est: 288.5]	Low Value [Est]	High Value [Est]
MEMC	2.0	0.9 [274.4]	3.0 [301.3]
TIMC	6.0	5.0 [274.0]	6.9 [301.6]
RTIM	6.0	4.7 [273.8]	7.2 [302.1]

** All of the "UTIL" variables are positively correlated with cost

Table 21

"UTIL" Changes as a Group: Drive Cost in Same Direction
(Moderate Project)

Variable	-9%	-8%	Baseline	+7%	+8%
MEMC	1.82	1.84	2.0	2.14	2.16
TIMC	5.46	5.52	6.0	6.42	6.48
RTIM	5.46	5.52	6.0	6.42	6.48
ESTIMATE	272.50	274.30	288.5	301.30	303.20

Complex Project. Changes in the variables were looked at individually and as a group. The results of individual changes in the CPLX1 related variables are listed in Table 22. Considering the increased complexity of this software program, smaller changes in variables cause larger changes in cost. Increments of 0.1 can cause greater variability and make it more difficult to reach the

Table 22

Ranges for Changes in Individual "CPLX1" Variables
(Complex Project)

Variable	Baseline Value [Est: 1055.5]	Low Value [Est]	High Value [Est]
# ACAP	5.5	6.1 [1003.8]	5.0 [1100.5]
# AEXP	1.0	1.6 [1005.1]	0.7 [1093.5]
# MODP	3.0	4.4 [1002.6]	1.7 [1107.0]
# PCAP	5.5	6.1 [1007.5]	4.9 [1105.7]
# TOOL	5.2	6.6 [1002.6]	3.9 [1107.0]
+ RCHG	7.0	6.0 [1004.1]	8.0 [1107.3]
+ HOST	5.1	4.5 [1007.4]	5.7 [1103.9]
# LEXP	1.0	1.8 [1003.6]	0.7 [1097.9]
+ SYST	2.0	1.7 [1010.3]	2.3 [1105.0]
# SEXP	1.0	1.2 [1015.9]	0.8 [1107.7]

Negatively correlated -- estimates decrease as variables increase

+ Positively correlated -- estimates increase as variables increase

extremes of the intervals. For example, a value for SEXP of 1.2 results in an estimate of 1015.9. A value of 1.3 causes the estimate to fall below the lower bound of 1002.3. The variables have the same relative impacts on cost (sensitivities) that they did for the simple and moderate projects.

Changes in the variables as a group were looked at next. Table 23 shows the changes that drive cost in the

Table 23

"CPLX1" Changes as a Group: Drive Cost in Same Direction
(Complex Project)

	Variable	-2.5%	-2%	Baseline	+1.5%	+2%
	ACAP	5.36	5.39	5.5	5.58	5.61
	AEXP	0.98	0.98	1.0	1.02	1.02
	MODP	2.93	2.94	3.0	3.05	3.06
	PCAP	5.36	5.39	5.5	5.58	5.61
	TOOL	5.07	5.10	5.2	5.28	5.30
*	RCHG	7.18	7.14	7.0	6.90	6.86
*	HOST	5.23	5.20	5.1	5.02	5.00
	LEXP	0.98	0.98	1.0	1.02	1.02
*	SYST	2.05	2.04	2.0	1.97	1.96
	SEXP	0.98	0.98	1.0	1.04	1.04
	ESTIMATE	1127.90	1106.90	1055.5	1012.60	1001.60

* Changes take opposite signs

same direction, while Table 24 shows the changes that involve trade-offs. Table 23 indicates that the variables can change by approximately 2% in both directions (for decreases: +2% for negatively correlated variables and -2% for positively correlated variables, and vice versa) without exceeding bounds. Table 24 shows that the "CPLX1" variables can increase and decrease between 8 and 9% without going over the limits.

Table 24

"CPLX1" Changes as a Group: Trade-Offs
(Complex Project)

Variable	-9%	-8%	Baseline	+8%
ACAP	5.00	5.06	5.5	5.94
AEXP	0.91	0.92	1.0	1.08
MODP	2.73	2.76	3.0	3.24
PCAP	5.00	5.06	5.5	5.94
TOOL	4.73	4.78	5.2	5.62
RCHG	6.37	6.44	7.0	7.56
HOST	4.64	4.69	5.1	5.51
LEXP	0.91	0.92	1.0	1.08
SYST	1.82	1.84	2.0	2.16
SEXP	0.91	0.92	1.0	1.08
ESTIMATE	1112.70	1105.40	1055.5	1004.20

The same procedure was followed for the UTIL r variables. Table 25 gives the results of changes i individual variables, while Table 26 gives the resu the changes in the variables as a group. Table 25 shows that the changes in the individual "UTIL" are comparable to the changes allowed in the simpl moderate projects, and TIMC, MEMC, and RTIM have th relative impacts as they did before. Table 26 indi that the "UTIL" variables can only decrease by slig than 6 percent, and increase by slightly less than

Table 25

Ranges for Changes in Individual "UTIL" Variables
(Complex Project)

Variable	Baseline Value [Est: 1055.5]	Low Value [Est]	High Value [Est]
MEMC	5.5	4.3 [1005.8]	6.7 [1105.5]
TIMC	7.0	6.0 [1004.7]	8.0 [1106.6]
RTIM	7.0	5.7 [1003.6]	8.3 [1107.8]

** All of the "UTIL" variables are positively correlated with cost

Table 26

"UTIL" Changes as a Group: Drive Cost in Same Direction
(Complex Project)

Variable	-6.5%	-6.0%	Baseline	+5.5%	+6.0%
MEMC	5.14	5.17	5.5	5.80	5.83
TIMC	6.55	6.58	7.0	7.39	7.42
RTIM	6.55	6.58	7.0	7.39	7.42
ESTIMATE	1000.60	1004.40	1055.5	1104.20	1108.30

without exceeding the bounds. As was the case with the simple and moderate projects, the domains of consistency can be derived from the tables.

Very Complex Project. Individual changes in the "CPLX1" variables were looked at first and the results are listed in Table 27. In comparison to the earlier projects, the individual variables cannot vary as much, without exceeding the bounds. The variables have the same relative

impacts on cost as they did for the other three projects. That is, the model is more sensitive to HOST, ACAP, PCAP, RCHG, and SEXP, than it is to SYST, LEXP, MODP, TOOL, and AEXP (rank order is maintained).

Table 27

Ranges for Changes in Individual "CPLX1" Variables
(Very Complex Project)

Variable	Baseline Value [Est: 4706.6]	Low Value [Est]	High Value [Est]
# ACAP	5.5	6.1 [4476.2]	5.0 [4907.6]
# AEXP	1.0	1.6 [4482.0]	0.7 [4876.1]
# MODP	5.2	6.5 [4487.4]	3.9 [4936.5]
# PCAP	5.5	6.1 [4492.7]	4.9 [4930.7]
# TOOL	7.5	8.8 [4487.4]	6.2 [4936.5]
+ RCHG	10.0	8.9 [4482.0]	* max [4706.6]
+ HOST	6.9	6.2 [4482.1]	7.6 [4932.9]
# LEXP	0.5	0.7 [4521.7]	0.3 [4934.2]
+ SYST	3.0	2.7 [4484.0]	* max [4706.6]
# SEXP	0.5	0.6 [4541.2]	0.4 [4888.7]

Negatively correlated -- estimates decrease as variables increase

+ Positively correlated -- estimates increase as variables increase

* Maximum values -- RCHG = 10.0 and SYST = 3.0

Changes in the "CPLX1" variables as a group were looked at next. Table 28 shows the changes that drive cost in the

same direction, while Table 29 shows the changes that involve trade-offs. Table 28 indicates that the variables can change by approximately 1.75% (+1.75% for negatively correlated variables and -1.75% for positively correlated variables) without the resulting estimate falling below the

Table 28

"CPLX1" Changes as a Group: Drive Cost in Same Direction
(Very Complex Project)

	Variable	-2.25%	-2.0%	Baseline	+1.5%	+2.0%
	ACAP	5.38	5.39	5.5	5.60	5.61
	AEXP	0.98	0.98	1.0	1.02	1.02
	MODP	5.08	5.10	5.2	5.29	5.30
	PCAP	5.38	5.39	5.5	5.60	5.61
	TOOL	7.33	7.35	7.5	7.63	7.65
*	RCHG	+ max	+ max	10.0	9.83	9.80
*	HOST	7.06	7.04	6.9	6.78	6.76
	LEXP	0.49	0.49	0.5	0.51	0.51
*	SYST	+ max	+ max	3.0	2.95	2.94
	SEXP	0.49	0.49	0.5	0.51	0.51
	ESTIMATE	4945.90	4920.10	4706.6	4451.80	4420.70

* Changes take opposite signs

\+ RCHG and SYST cannot be increased above their maximums

Table 29

"CPLX1" Changes as a Group: Trade-Offs
(Very Complex Project)

Variable	-15.5%	-15.0%	Baseline	+4.0%	+4.5%
ACAP	4.65	4.68	5.5	5.72	5.75
AEXP	0.85	0.85	1.0	1.04	1.05
MODP	4.39	4.42	5.2	5.41	5.43
PCAP	4.65	4.68	5.5	5.72	5.75
TOOL	6.34	6.38	7.5	7.80	7.84
RCHG	8.45	8.50	10.0	* max	* max
HOST	5.83	5.87	6.9	7.18	7.21
LEXP	0.42	0.43	0.5	0.52	0.52
SYST	2.54	2.55	3.0	* max	* max
SEXP	0.42	0.43	0.5	0.52	0.52
ESTIMATE	4958.10	4935.90	4706.6	4473.60	4446.90

* Cannot increase RCHG and SYST above their baseline values

lower bound; and, can change by approximately 2.25% (-2.25% for inverse variables and +2.25% for direct variables), without exceeding the upper bound. Table 29 shows that the "CPLX1" variables can all increase by 4% without the estimate falling below the lower bound, and can decrease between 15 and 15.5% without the estimate exceeding the upper bound.

The same procedure was followed for the UTIL related variables. Table 30 gives the results of changes in individual variables, while Table 31 gives the results of the changes in the variables as a group that drive cost in the same direction. Table 30 shows that the changes in the individual "UTIL" variables are comparable to the changes allowed in the simple, moderate, and complex projects (i.e., same rank order in terms of the model's sensitivity to them). Table 31 indicates that the "UTIL" variables can only increase and decrease by slightly more than 5.5%, without exceeding the bounds. As before, the domains of consistency can be derived from the tables.

Table 30

Ranges for Changes in Individual "UTIL" Variables
(Very Complex Project)

Variable	Baseline Value [Est: 4706.6]	Low Value [Est]	High Value [Est]
MEMC	6.0	4.8 [4488.8]	7.3 [4944.4]
TIMC	8.0	7.0 [4489.0]	9.0 [4925.9]
RTIM	8.0	6.7 [4482.3]	9.3 [4932.8]

** All of the "UTIL" variables are positively correlated with cost

Table 31

"UTIL" Changes as a Group: Drive Cost in Same Direction
(Very Complex Project)

Variable	-6.0%	-5.5%	Baseline	+5.5%	+6.0%
MEMC	5.64	5.67	6.0	6.33	6.36
TIMC	7.52	7.56	8.0	8.44	8.48
RTIM	7.52	7.56	8.0	8.44	8.48
ESTIMATE	4457.80	4478.20	4706.6	4943.20	4965.10

Comparison of Domains of Consistency

Before the domains of consistency are compared, a short review is appropriate. The domain of consistency contained those System-3 values for which the System-3 model produced estimates reasonably close (+/- 5%) to the estimates produced by the PRICE S model for a given software development effort. The intervals for each of the software development projects were as follows:

Simple - [52.3, 57.8]
Moderate - [273.6, 302.4]
Complex - [1002.3, 1107.3]
Very Complex - [4473.6, 4944.5]

Since most of the System-3 variables were set to specific values for each development effort, the domain of consistency was defined in terms of those System-3 variables that related to the PRICE S variables CPLX1 and UTIL.

For each of the different development efforts, there were a number of possible domains of consistency for the variables that were allowed to vary. For the "CPLX1"

variables, changes in individual variables, changes as a group that drove cost in the same direction, and changes as a group that allowed for trade-offs, were all considered as long as the resulting estimates were within the specified interval. For the "UTIL" variables, only the first two types of changes applied.

Table 32 summarizes the domains of consistency resulting from group changes that drove cost in one direction and those that allowed for trade-offs. It indicates the percentage amounts of change in the "CPLX1" and "UTIL" variables that can be made, such that the resulting estimates are within the intervals specified above. Estimates cannot fall below the lower bound nor exceed the upper bound.

The spread between changes in the negative direction and changes in the positive direction can be looked at to compare the domains of consistency for the four software development efforts. The spread in changes to drive cost down and to drive cost up is comparable across all four projects for the "CPLX1" variables. For the simple and moderate projects, that spread (a total of 5.5%) is only slightly higher than that for the complex and very complex projects (4.5%). On the other hand, the spread in changes allowed for trade-offs is not. It is smallest for the simple project (13), but more consistent for the other three (22, 18, 19.5).

Table 32

Domains of Consistency

	"CPLX1" Variables	"UTIL" Variables
Simple:		
Drive Cost Down	+3.5%*	-12.0%
Drive Cost Up	-2.0%*	+7.0%
Trade-offs (Decrease Cost)	+8.0%	N/A
Trade-offs (Increase Cost)	-5.0%	N/A
Moderate:		
Drive Cost Down	+3.0%*	-(8-9%)
Drive Cost Up	-2.5%*	+(7-8%)
Trade-offs (Decrease Cost)	+11.0%	N/A
Trade-offs (Increase Cost)	-11.0%	N/A
Complex:		
Drive Cost Down	+(1.5-2%)*	-(6-6.5%)
Drive Cost Up	-(2-2.5%)*	+(5.5-6%)
Trade-offs (Decrease Cost)	+(8-9%)	N/A
Trade-offs (Increase Cost)	-(8-9%)	N/A
Very Complex:		
Drive Cost Down	+(1.5-2%)*	-(5.5-6%)
Drive Cost Up	-(2-2.5%)*	+(5.5-6%)
Trade-offs (Decrease Cost)	+4.0%	N/A
Trade-offs (Increase Cost)	-(15-15.5%)	N/A

* RCHG, HOST, and SYST changes have opposite signs

For the "UTIL" variables, the spreads seem to be at one level for the simple and moderate projects (19, 17), and at another level for the complex and very complex projects (12.5, 12).

Table 33 presents the data from Table 32 in a different manner to allow for an easier comparison, when looking at changes to drive cost down, changes to drive cost up, trade-offs that decrease cost, and trade-offs that increase cost. Table 33 indicates that the changes allowed in the "CPLX1" variables to drive cost down or to drive cost up are fairly consistent across all four software development efforts. On the other hand, for the "UTIL" variables, the changes are larger for the simple and moderate efforts than for the complex and very complex efforts. This may indicate that a non-linear relationship exists. That is, the "UTIL" variables may have a greater impact on cost at higher levels than they do at lower levels.

By comparing the changes allowed in the "CPLX1" variables to the changes allowed in the "UTIL" variables, it is evident that the changes are greater for the "UTIL" variables than for the "CPLX1" variables. That occurs across all four software projects. This would seem to indicate that the model is more sensitive to the "CPLX1" variables than it is to the "UTIL" variables.

Table 33

Comparison of Domains of Consistency

	"CPLX1" Variables	"UTIL" Variables
Drive Cost Down:		
Simple Project	+3.5%*	-12.0%
Moderate Project	+3.0%*	-(8-9%)
Complex Project	+(1.5-2.0%)*	-(6.0-6.5%)
Very Complex Project	+(1.5-2.0%)*	-(5.5-6.0%)
Drive Cost Up:		
Simple Project	-2.0%*	+7%
Moderate Project	-2.5%*	+(7-8%)
Complex Project	-(2.0-2.5%)*	+(5.5-6.0%)
Very Complex Project	-(2.0-2.5%)*	+(5.5-6.0%)
Trade-offs (Decrease Cost)		
Simple Project	+8%	N/A
Moderate Project	+11%	N/A
Complex Project	+(8-9%)	N/A
Very Complex Project	+4%	N/A
Trade-offs (Increase Cost)		
Simple Project	-5%	N/A
Moderate Project	-11%	N/A
Complex Project	-(8-9%)	N/A
Very Complex Project	-(15.0-15.5%)	N/A

* RCHG, HOST, and SYST changes have opposite signs

For changes that allowed for trade-offs that decreased cost, the simple, moderate, and complex projects allowed relatively more change than the very complex project. On the other hand, for changes that allowed for trade-offs that

increased cost, the complex project changes are greater than those for the other three projects.

The approaches to changes in the variables can also be compared. The changes in individual variables cannot be overlooked, but very seldom is there a change in only one variable since there is interplay among the "CPLX1" and "UTIL" variables. The most appropriate domains of consistency to use for each software development effort are those resulting from changes in the "CPLX1" and "UTIL" variables as a group. Those domains contain input values that could reasonably be used to describe the corresponding software effort. Changes in the "CPLX1" variables that drive cost in the same direction imply either that the work is simpler (fewer changing requirements, less rehosting, and less complex development computing resources) and is performed by a more able team, or that the work is more difficult and is performed by a less able team. Although both situations are possible, the former situation is more desirable than than the latter.

On the other hand, the changes that allow for trade-offs imply that less difficult work is performed by a less able team, while more difficult work is performed by a more able team. These "trade-offs" represent an ideal use of resources. In conclusion, these domains of consistency contain input values that could reasonably be used to describe the corresponding software project.

V. Conclusions and Recommendations

The purpose of this research effort was to perform an analysis on the PRICE S and System-3 software cost estimating models, for four software developing projects, in order to determine the domains of consistency. That is, to determine the sets of inputs for which the two models provide approximately equal estimates. Because of an infinite number of possible variable combinations, no one specific domain of consistency can be specified. However, some generalizations can be made concerning the domain of consistency for each of the four software development efforts.

Conclusions

Before the domain of consistency could be determined, much preliminary work needed to be accomplished. Of the eight investigative questions presented in Chapter I, six had to be answered to lay the foundation for the sensitivity analysis, and subsequently, the determination of the domain of consistency. The findings for the investigative questions are discussed in relation to the research objectives which they satisfy.

Research Objective #1. "Determine the cost estimate produced by PRICE S for each of four software development efforts representing simple, moderate, complex, and very complex projects". To satisfy this objective, it was

necessary to examine the inputs required to run PRICE S, to determine reasonable values for those inputs to describe four, increasingly complex software development efforts, and to run the PRICE S model with those values to produce estimates for those efforts. The eleven main inputs required to run the PRICE S model were identified (refer to Table 2). For each of the four software development efforts, it was assumed that the software would require operation in a mil-spec avionics environment and that it would be developed by a typical ASD contractor. Therefore, two of the PRICE S variables (PLTFM and PROFAC) were held constant for all four efforts. The one PRICE S variable (CPLX2) that did not have a PRICE S equivalent was set to a value that minimized its impact on cost. The values chosen for the other input variables were based on possible values designated in the PRICE S Reference Manual (19). As the project increased in difficulty, the variables were set to appropriate values. The PRICE S constructs are strictly hypothetical. Finally, the PRICE S model was run and estimates were obtained. The input values and resulting estimates for the four projects were discussed in Chapter IV (refer to Table 9).

Research Objective #2. "Determine the domain of consistency, both individually and as a group, for selected inputs to the System-3 model when the criteria for consistency are based on the PRICE S model". To satisfy

this objective, the System-3 variables were examined and then compared to the PRICE S variables. There are 24 technology and environment input variables, as well as four additional variables (refer to Table 3). Next, the relationship between the System-3 inputs and the PRICE S inputs was determined. A summary of those relationships was presented in Table 8. There were some System-3 inputs that had one-to-one correspondence with PRICE S variables, some that when combined with other System-3 variables corresponded to a single PRICE S variable, and a few that had no PRICE S equivalent.

A number of the System-3 values were set to specific values. Since PLTFM and PROFAC were constant for all four software development efforts, the System-3 variables that correspond to those variables were also held constant. For each effort, the System-3 variables that have one-to-one relationships with PRICE S variables were set to appropriate System-3 values and were not allowed to vary. Additionally, the System-3 variables that have no PRICE S equivalent were set to values that minimized their impacts on cost. The only variables that were allowed to vary were those System-3 variables that relate to the PRICE S variables CPLX1 and UTIL.

The domain of consistency for the CPLX1 and UTIL related variables were considered separately and could be specified in terms of changes in individual variables or

changes as a group. The domains of consistency for the changes in dindividual variables is represented by the initial set of System-3 values, except for the changed variable which can take on a range of values. The domains of consistency for each software development effort are derived from Tables 12, 17, 22, and 27 (for the "CPLX1" variables) and from Tables 15, 20, 25, and 30 (for the "UTIL" variables). The domains of consistency for group changes that drive cost in one direction and for group changes that allow for trade-offs are summarized in Table 32.

Research Objective #3. "Determine if each domain of consistency includes input values that could reasonably be used to describe the corresponding software project". This research objective was satisfied by examining the domains of consistency resulting from the sensitivity analyses. Each domain of consistency for the CPLX1 and UTIL related variables does contain reasonable input values for the related software development effort. However, the most appropriate domains of consistency to use for the CPLX1 related variables are those resulting from changes as a group that allow for trade-offs. Those changes imply that less difficult work is performned by a less able team, while more difficult work is performed by a more able team. That is the way that work typically gets assigned in an organization.

Recommendations for Further Research

There are three areas in which similar analysis can be applied. The following areas would benefit from more research:

1. Since this research used the project level, expand the analysis to consider component levels.

2. Use real world data rather than hypothetical data for the PRICE S estimates.

3. Apply this type of analysis to different software cost estimating models.

The field of software cost estimating and related models is still wide open and worthy of much more research.

Appendix A: PRICE S Variables

The material in this appendix is taken from the PRICE S Reference Manual (19).

> PLTFM [Platform] is the variable which describes the customer's requirements stemming from his planned operating environment. It is a measure of the portability, reliability, structuring, testing and documentation required for acceptable contract performance. The key to selecting values for the PLTFM variable is not so much location of the software, but the specification which must be met. Typical values are summarized below. Although discrete values are indicated, it should be emphasized that PLTFM is a continuous variable ranging from 0.6 to 2.5 [19:II-A2].

Typical Platform Values (19:II-A2)

Operating Environment	PLTFM
Production Center Internally Developed S/W	0.6-0.8
Production Center Contracted S/W	1.0
MIL-Spec Ground	1.2
Military Mobile (Van or Shipboard)	1.4
Commercial Avionics	1.7
MIL-Spec Avionics	1.8
Unmanned Space	2.0
Manned Space	2.5

> CPLXM [Management Complexity] is the variable which provides a quantitative description of the relative effect of complicating factors on the software task. The factors entering into CPLXM are multinational projects and/or software developed at more than one location. The scale for CPLXM is normalized such that a value of 1.0 represents a general defense industry average. Typical adjustments to CPLXM which, when applied to a basic CPLXM of 1.0, yield appropriate values, are listed below [19:II-A2, II-A3].

Typical Management Complexity Adjustment (19:II-A3)

	CPLX Adjustment
More Than One Location	+0.2
Multinational Project	+0.2
More Than One Location and Multinational Project	+0.4

> UTIL [Utilization] is the fraction of available hardware cycle time or total memory capacity used. It describes the extra effort needed to adapt software to operate within limited processor capabilities. Recent technological advances are rapidly changing the relative importance of memory capacity and speed. Historically, memory limitations were frequently severe, requiring skill and reprogramming to make efficient use of all available memory space. The advent of relatively inexpensive, expandable memories has improved this limitation in many applications. Use of available machine cycle time is often a more critical factor.
>
> When available storage space is 70% utilized by data storage requirements, UTIL = 0.7. When program execution requires 80% of the machine cycle time at peak load, UTIL = 0.8. The larger of the two values normally governs the selection of the UTIL values to be entered. UTIL ranges from 0.0 to 10.0 [19:II-A4].

LANG (Language) is an "entry which consists of a keyword, 'LANG=', and the name of the source language to be used for the software development effort" (19:II-A6). The possible source languages are as follows:

Ada	IFAM
ALGOL	JOVIAL
APL	MICROCODE
ASSEMBLY	PASCAL
ATLAS	PL1
BASIC	PRIDE
CMS2	SPL1
COBOL	C
COMPASS	HIGHER ORDER
CORAL66	MACHINE
FLOD	4TH GENERATION
FORTRAN	INTERPRETATIVE

Higher order, machine, 4th generation, and interpretative are used if the language to be used is not specifically listed (19:II-A6, II-A7).

"SLOC (Source Lines of Code) is the total number of source lines of code to be developed. Comments imbedded in the code are not to be included" (19:II-A7).

> CPLX1 [Complexity] is the variable which provides a quantitative description of the relative effect of complicating factors on the software development task. Factors entering into CPLX1 are product familiarity, personnel skills, software tools, and any unusual factors present in the development environment that affect the development schedule. Since the above factors control development time, CPLX1 is directly relatable to performance schedule [19:II-A8].
>
> The scale for CPLX1 is normalized such that a value of 1.0 represents a general defense industry average of a "normal" crew working on a "normal", new project. [The table below] shows typical CPLX1 adjustments which, when applied to a basic CPLX1 value of 1.0 yield appropriate values [19:II-A8].

Typical Complexity Adjustments (19:II-A9)

	CPLX1 Adjustment
Personnel	
Outstanding Crew, Among Best in Industry	-0.2
Extensive Experience, Some Top Talent	-0.1
Normal Crew, Experienced	0
Mixed Experience, Some New Hires	+0.1
Relatively Inexperienced, Many New Hires	+0.2
Product Familiarity	
Familiar Type of Project	-0.1
Normal New Project	0
New Line of Business	+0.2
Software Tools	
Very High	-0.2
High	-0.1
Nominal	0
Low	+0.1
Very Low	+0.2
Complicating Factors	
New Language	+0.2
Many Changing Requirements	+0.2

CPLX2 [HSI Complexity] is the variable which provides a quantitative description of the relative effect of complicating factors on the software development task caused by hardware/software interactions. Factors entering into CPLX2 include new hardware development and hardware developed in parallel. The scale for CPLX2 is normalized such that a value of 1.0 represents a general defense industry average of a "normal" crew working on a "normal", new project. [The table below] shows typical CPLX2 adjustments which, when applied to a basic CPLX2 value of 1.0 yield appropriate values [19:II-A8, II-A10].

Typical HSI Complexity Adjustments (19:II-A10)

	CPLX2 Adjustment
No Complicating Factors	0
New Hardware	+0.2
Hardware Developed in Parallel	+0.3

PROFAC [Productivity Factor] is an empirically derived parameter which includes such items as skill levels, experience, productivity, and efficiency. The combined effect of such factors tends to remain constant within a particular organization. PROFAC thus provides a mechanism for consolidating tendencies on software development costs [19:II-A10].

Typical Productivity Factors (19:II-A11)

Environment	PROFAC
Commercial Applications: Typically represented by APPL values between 2.0 and 4.0, PLTFM values of 1.0 or less, and implementation in languages such as FORTRAN, COBOL, BASIC, etc.	8.0 - 4.0
Aerospace Applications: Typically represented by APPL values between 3.0 and 9.0, PLTFM values of 1.2 or greater, and implementation is languages such as FORTRAN, BASIC, PASCAL, ATLAS, Ada, ASSEMBLY, CMS-2, C, PL1, ALGOL, JOVIAL (J73/J3B), etc.	5.0 - 3.0

APPL [Application] is the user defined application value than corresponds to the MAPP value. APPL is the parameter that summarizes the application mix of source lines of code. Its normal range is from 0.866 to 10.952. Values toward the lower end of the range describe programs that are predominantly Math and String Manipulation. Values toward the higher end represent greater emphasis on Real Time Command and Control and Interactive Operations. In this sense, APPL represents an inherent complexity, independent of variations in schedule, Productivity Factor, operating environment, and system utilization [19:II-A12].

APPL Mix Elements (19:II-A13)

Instruction Mix	Application
Operating Systems (MOPR)	10.95
Interactive Operations (MINT)	10.95
Real Time Command and Control (MREA)	8.46
On-Line Communications (MONL)	6.16
Data Storage and Retrieval (MDAT)	4.10
String Manipulation (MSTR)	2.31
Mathematical Operations (MMAT)	0.86

NEW DESIGN is the amount of new design required for the application. NEW CODE is the amount of new code required for the application.

Appendix B: System-3 Variables

Developer Technology

Application (APPL) class complexity rating establishes the overall level of system application (4:7-24):

Low	1.0	Business Data Processing
	1.3	Simple Signal Processing
	1.5	Interface Systems
Nominal	2.0	Applications with Complex Systems, File and/or User Interfaces, Such as Command and Control
	2.0	Communication Networks
	2.5	Highly Complex Command & Control Systems
High	3.0	Computer Networks, Operating Systems, Compilers, etc.
	3.0	Fire Control Systems
	3.0	Complex Signal Acquisition & Processing

Analyst capability (ACAP) measures capability of the analyst software team assigned to a specific project over the duration of the project (4:7-2):

Very Low	1.5	Non-Functional Team
Low	3.5	Functional with Low Affectivity
Nominal	5.5	Functional & Effective
High	7.5	Extraordinary
Very High	9.0	Near Perfect Functioning Team

Application experience (AEXP) is the project team's equivalent experience level with this general type of application (4:7-4):

Very Low	0.3	< 4 Months
Low	1.0	1 Year
Nominal	3.0	3 Years
High	6.0	6 Years
Very High	12.0	12 or More Years or Re-implementation By the Same Team

Modern Development Practices (MODP) rates the developer's use of modern software development methods and practices (4:7-25, 7-26):

Very Low	0.0	No Use of MODP
Low	3.0	Beginning Experimental Use
Nominal	5.2	Reasonably Experienced in Some Practices
High	7.5	Reasonably Experienced in Most Practices
Very High	10.0	Routine Use of All Practices

Programmer capabilities (PCAP) rates the ability of the programming team to function effectively (4:7-30):

Very Low	1.5	Non-Functioning Team
Low	3.5	Functional but Not Very Effective
Nominal	5.5	Functional and Effective
High	7.5	Extraordinary
Very High	9.0	Nearly Perfect

Tool support, automated (TOOL) rates the support of development practices with automated software development tools (4:7-58):

Very Low	0.0	Primitive Tools (Bit Switches, Dumps)
Low	3.0	Base Batch Tools, 370 OS Type
Nominal	5.2	Interactive, PWB UNIX
High	7.5	Modern, Ada Minimal APSE
Very High	10.0	Advanced, Full Ada APSE Ada

Turnaround time (TURN) is the number of hours of wasted time from logging onto the development system until hard copy output is obtained (4:7-61):

Very Low	0.1	Less Than 6 Minutes
Low	0.5	Turnaround 30 Minutes
Nominal	2.0	Turnaround 2 Hours
High	4.0	Turnaround 4 Hours
Very High	8.0	Turnaround 8 Hours

Environmental - Computer

Special display requirements (DISP) rates the extra effort required to bullet proof an respond to changing user needs (4:7-12, 7-13):

Nominal	0.0	Simple Inputs/Outputs
High	3.2	User Friendly Error Recovery and Menus
Very High	6.8	Interactive
Extra High	10.0	Complex (Such as CAD/CAM)

Memory constraints (MEMC) measure the probable amount of extra effort required to fit the software into the target machine (4:7-22):

Nominal	0.0	No Memory Constraints
High	1.0	Some Overlaying or Segmentation
Very High	4.0	Extensive Overlaying or Segmentation
Extra High	10.0	Complex Memory Management and Economic Measures

Time constraints code (TIMC) is code that must have special attention to ensure adequate time performance (not real time) (4:7-57):

Nominal	0.0	No Time Constraints
High	2.5	25% of Code is Time Constrained
Very High	5.0	50% of Code is Time Constrained
Extra High	7.5	75% of Code is Time Constrained

Real time operation (RTIM) rates the amount of source code that must actually handle externally timed operations (4:7-34):

Nominal	0.0	0% of Source Lines Real Time
High	2.5	25% of Source Lines with Real Time Constraints
Very High	5.0	50% of Source Lines with Real Time Constraints
Extra High	10.0	100% of Source Lines with Real Time Constraints

Environmental - Product

Specification level (SPEC) rates the required design, code, test, and other documentation level (4:7-47):

Low	0.0	Personal Software User Developed
Nominal	3.0	Commercial Software User Developed
High	4.0	Mil-Spec Complete Essential Documentation (Mil 483/490)
Very High	6.0	Mil-Spec Full Documentation Commercial Software Contractor Developed (DoD 2167)
Extra High	10.0	High Reliability - Public Safety Requirements

Quality assurance level (QUAL) is the depth with which quality assurance will be used for the software (4:7-37):

Very Low	0.0	No Quality Assurance
Low	1.6	Informal Quality
Nominal	4.0	Mil-Spec QA
Very High	6.0	QA for High Potential Loss
Extra High	10.0	Risk of Loss to Human Life

Test level (TEST) is the depth at which the software will undergo testing by the developer organization (4:7-56):

Very Low	0.0	No Formal Testing
Low	1.6	Informal Testing by Developer
Nominal	4.0	Normal Testing by Developer
Very High	6.0	High Potential Loss Testing
Extra High	10.0	Risk of Loss to Human Life

Requirements change (RCHG) rates the level of anticipated changes to the requirements throughout the development cycle (after baseline) (4:7-42):

Low	0.0	Essentially No Requirement Changes
Nominal	1.3	Small Non-Critical Redirections
High	4.0	Occasional Moderate Redirections
Very High	7.0	Frequent Moderate & Occasional Major Changes
Extra High	10.0	Frequent Moderate & Frequent Major Changes

Rehosting (HOST) is movement from original development facilities to other machines, compilers, programming languages, etc. (4:7-35):

Nominal	0.0	No Rehosting
High	3.3	Minor Language and/or System Change
Very High	6.7	Major Language or System Change
Extra High	10.0	Major Language and System Change

Language type rating (LANG) is the primary language to be used during development (4:7-17):

Low	0.5	BASIC, 4GL
Nominal	1.0	Algol, Assembler (all), C, COBOL, Forth, FORTRAN, Pascal, PL/M
High	2.0	CMS-2, Jovial, Modula 2, PL/1-G
Very High	3.0	Ada, PL/1-F

Language experience (LEXP) rates the teams experience with similar programming languages at the start of Full Scale Development (4:7-18):

Extra Low	0.1	Less than 1 Month
Low	0.3	4 Months Average
Nominal	1.0	1 Year Average Experience
High	3.0	3 Years Average Experience
Very High	5.0	5 Years Average Experience

System complexity - virtual (SYST) rates the inherent difficulty in learning and richness of the development computing resources (4:7-51):

Low	1.0	Simple Virtual Machines, IBM PC, CP/M
Nominal	2.0	DEC VAX 11/780 VMS, IBM 370 DOS, UNIX
High	3.0	Ada Programming Support Environment (APSE)

System experience - virtual (SEXP) rates the level of experience the developers have with the virtual machine at the start of full scale development (4:7-52):

Very Low	0.1	< 1 Month Experience
Low	0.3	4 Months Experience
Nominal	1.0	1 Year
High	3.0	3 Years

Virtual machine volatility (VMVL) rates the rate of changes within the development computing resources (4:7-63):

Low	0.0	No Major Changes, Minor Each Year
Nominal	2.5	Major Changes Each 12 Months, Minor Each Month
High	4.6	Major Changes Each 6 Months, Minor Each 2 Weeks
Very High	7.2	Major Changes Each 2 Months, Minor Each Week
Extra High	10.0	Major Changes Each 2 Weeks, Minor Each 2 Days

Environmental - Support

Multiple site development (MULT) rates the inefficiency due to site and organizational diversity (4:7-20):

Nominal	0.0	Single Site & Single Organization
High	3.4	Single Site & Multiple Org.
Very High	6.5	Multiple Sites, Same General Location (< 30 Minutes)
Extra High	10.0	Multiple Sites, 50 or More Miles Apart

Resource dedication (RDED) rates the availability of development computers and other computer resources (4:7-44):

Extra Low	1.0	10% Access to Computing Resources
Very Low	4.0	40% Access to Computing Resources
Low	7.0	70% Access to Computing Resources
Nominal	10.0	Fully Dedicated

Resource and support location (RLOC) rates the wasted time accessing support personnel and tools by the developers (4:7-45):

Nominal	0.0	Local Development Resources and Support
High	1.5	50 Mile Radius or 1.5 Hours
Very High	5.0	200 Mile Radius or 5 Hours
Extra High	10.0	400 Mile Radius of Greater

Size/Complexity

Lines of code (LOC) are non-comment source lines including: executable logic and computation, data declaration, input/output, and deliverable job control (4:7-19).

Complexity specifies the software component's difficulty independent of the developer's ability to implement the component (4:7-8):

Extra High	4.0	Development primarily using micro-code for the application. Signal processing systems with extremely complex interfaces and control logic.
	7.0	99% Mil-Std Software (At Task Level) No More Complex Than 7
Very High	8.0	New systems with significant interaction requirements with larger system structure. Operating system & real-time with significant logical code.
High	11.0	Application program with signifi-cant logical complexity. Some changes to operating system, but minor real-time processing or special display hardware require-ments.
Nominal	15.0	New stand alone systems developed on firm operating systems. Minimal interface problems with underlying operating system or other system parts.
	15.0	99% Mil-Std Software (At Task Level) No More Simple
Low	21.0	Software of low logical complexity using straight forward I/O and primarily internal data storage.
Very Low	28.0	Extremely simple software with primarily straight line code, simple I/O, and internal storage arrays.

New disign is the amount of new design required for the application. New code is the amount of new code required for the application.

Appendix C: PRICE S and System-3 Output for Simple Project

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

SIMPLE PROJECT — Development Item

ITEM DESCRIPTORS

Platform	1.8	Mgmt Complexity	1.00	Cost	0.0
Internal Integ	0.50	External Integ	0.50	Utilization	0.50

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Specification Review	188
Preliminary Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language FORTRAN		Source Code	10000	Non-executable SLOC	0.00
Complexity 1	0.90	Complexity 2	1.00	Productivity Factor	4.00

Application Categories	Mix	New Design	New Code
User Defined (APPL= 4.10)	1.00	0.40	0.40
Data S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Opr Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

SIMPLE PROJECT Development Item

COSTS IN PERSON MONTHS

	Design	Prog	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	0.	0.	0.	0.	0.	0.	0.
Sys/SW Reqt	0.	0.	0.	0.	0.	0.	0.
SW Requirement	0.	0.	0.	0.	0.	0.	0.
Prelim Design	6.	1.	0.	0.	1.	1.	10.
Detail Design	6.	4.	0.	0.	1.	1.	12.
Code/Test	5.	5.	0.	0.	1.	1.	12.
CSCI Test	7.	9.	0.	0.	2.	3.	21.
System Test	0.	0.	0.	0.	0.	0.	0.
Oper T & E	0.	0.	0.	0.	0.	0.	0.
TOTAL	24.	19.	0.	0.	6.	6.	55.

SCHEDULE INFORMATION

Concept Start	AUG 87*		TRR	JUL 88*	(2.0)
SRR	SEP 87*	(1.5)	FCA	NOV 88*	(4.2)
SDR	OCT 87*	(1.0)	PCA	DEC 88*	(1.2)
SSR	JAN 88	(3.1)	FQR	FEB 89*	(1.2)
PDR	MAR 88*	(2.0)	OTE	MAY 89*	(3.0)
CDR	MAY 88*	(2.0)			

SUPPLEMENTAL INFORMATION

Source Lines of Code	10000
Source Lines of Code/Person Month	181.6

CEI-3 Schedule and Cost Estimation System

SIMPLE PROJECT
Current Input Parameters
Task: SIMPLE

	Minimum	Nominal	Maximum	Mean	Std Dev
DEVELOPER TECHNOLOGY					
APPLication Complexity . . .	3.0	3.0	3.0	3.0	0.0
Analyst CAPability	3.5	3.5	3.5	3.5	0.0
Application EXPerience . . .	1.0	1.0	1.0	1.0	0.0
MODern Practices, use of . .	3.0	3.0	3.0	3.0	0.0
Programmer CAPability . . .	3.5	3.5	3.5	3.5	0.0
TOOL support, automated . .	3.0	3.0	3.0	3.0	0.0
TURNaround, logon-hardcopy .	2.0	2.0	2.0	2.0	0.0
ENVIRONMENTAL - COMPUTER					
DISP req'mt, special	6.8	6.8	6.8	6.8	0.0
MEMory Constraint	1.0	1.0	1.0	1.0	0.0
TIMe Constraint	5.0	5.0	5.0	5.0	0.0
Real TIMe	5.0	5.0	5.0	5.0	0.0
ENVIRONMENTAL - PRODUCT					
SPECification Level	6.0	6.0	6.0	6.0	0.0
QUALity Assurance Level . .	4.0	4.0	4.0	4.0	0.0
TEST Level	4.0	4.0	4.0	4.0	0.0
Requirements CHanGe vol . .	1.3	1.3	1.3	1.3	0.0
ReHOSTing develop->target .	1.0	1.0	1.0	1.0	0.0
LANGuage type rating	1.0	1.0	1.0	1.0	0.0
Language EXPerience	1.0	1.0	1.0	1.0	0.0
SYSTem complexity-virtual .	2.0	2.0	2.0	2.0	0.0
System EXPerience-virtual .	1.0	1.0	1.0	1.0	0.0
Virtual Mach. VoLatility . .	0.0	0.0	0.0	0.0	0.0
ENVIRONMENTAL - SUPPORT					
MULTiple site development .	0.0	0.0	0.0	0.0	0.0
Resource DEDication	10.0	10.0	10.0	10.0	0.0
Resource/support LOCation .	0.0	0.0	0.0	0.0	0.0
SIZE & COMPLEXITY SUMMARY					
New Lines of Code	4000	4000	4000	4000	0
Existing Lines of Code . . .	6000	6000	6000	6000	0
Lines to be Deleted	0	0	0	0	0
Lines to be Modified	0	0	0	0	0
Complexity	15.0	15.0	15.0	15.0	0.0

Current Input Parameters (Continued)

REUSE - REBUILD IMPACT	
% Design Effort Needed . . .	40.0
% Implement Eff. Needed . .	40.0
% Testing Effort Needed . .	40.0

FINANCIAL FACTORS	
Average Staff Pay Rate . . .	10000.0
Target Schedule	0.0
% Requirements Effort . . .	0.0
Req's Eff. Complete @ C/A .	0.0
Req's Schedule Constraint .	0.0
% Integration Effort	0.0
Avg. Annual Inflation . . .	0.0

SIMPLE PROJECT
Summary Report
Task : SIMPLE

(Minimum Time) Estimate

FULL SCALE DEVELOPMENT			
Development Time	11.03	0.00	Months
Development Effort	55.89	0.00	Person Months
Project Staff, Peak	7.68		Persons
Actual Staffing Rate	13.79	0.00	Persons/Year
Productivity	71.57		Lines/Staff/Month
Cost per Source Line	139.74		Dollars/Line
Total Full Scale Dev Task Cost	558.94		K-Dollars
REQUIREMENTS and INTEGRATION			
Requirements Time	0.00		Months
Requirements Effort	0.00		Person Months
Total Requirments Cost	0.00		K-Dollars
Integration Time	0.00		Months
Integration Effort	0.00		Person Months
Total Integration Cost	0.00		K-Dollars
Complexity	15.00	0.00	
Basic Technology Rating	3428.41	0.00	
Effective Technology Rating	1275.47	0.00	
Average Annual Inflation	0.00		
Effective Task Size	4000	0.0	Source Lines
Total Task Size	10000		Source Lines

SIMPLE PROJECT
Milestone Report
Task: SIMPLE

Milestone	Date (month)	Reqmts Effort (pmos)	Develop Effort (pmos)	Total Effort (pmos)
C/A	0.0	0.0	0.0	0.0
SDR	0.0	0.0	0.0	0.0
PDR	2.3	0.0	3.1	3.1
CDR	4.7	0.0	12.5	12.5
CUT	6.9	0.0	25.5	25.5
FQT	11.0	0.0	55.9	55.9
DT&E	11.0	0.0	55.9	55.9

Development Cost Spread Matrix
Allocated by Person Months
Part One
Task: SIMPLE

Phase	System Eng.	Proj. Mgmt.	Design	Programmers
C/A	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0
SDR - PDR	0.0	0.0	2.9	0.1
PDR - CDR	0.0	0.4	7.4	0.8
CDR - CUT	0.0	0.6	1.5	7.6
CUT - FQT	0.0	2.4	4.6	8.6
SUBTOTAL	0.0	3.4	16.4	17.2
DT&E	0.0	0.0	0.0	0.0
Totals	0.0	3.4	16.4	17.2

Part Two

Phase	Q.A.	C.M.	Test	Data Manip.	Total
C/A	0.0	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0	0.0
SDR - PDR	0.0	0.0	0.0	0.0	3.1
PDR - CDR	0.2	0.1	0.3	0.2	9.4
CDR - CUT	0.6	0.5	1.6	0.5	13.0
CUT - FQT	1.7	1.7	10.8	0.5	30.4
SUBTOTAL	2.6	2.3	12.7	1.3	55.9
DT&E	0.0	0.0	0.0	0.0	0.0
Totals	2.6	2.3	12.7	1.3	55.9

Appendix D: PRICE S and System-3 Output for Moderate Project

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

MODERATE COMPLEXITY PROJECT — Development Item

ITEM DESCRIPTORS

Platform	1.8	Mgmt Complexity	1.00	Cost	0.0
Internal Integ	0.50	External Integ	0.50	Utilization	0.60

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Specification Review	188
Preliminary Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language PASCAL		Source Code	25000	Non-executable SLOC	0.00
Complexity 1	1.10	Complexity 2	1.00	Productivity Factor	4.00

Application Categories	Mix	New Design	New Code
User Defined (APPL= 6.16)	1.00	0.50	0.50
Data S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Opr Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

MODERATE COMPLEXITY PROJECT — Development Item

COSTS IN PERSON MONTHS

	Design	Prog	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	0.	0.	0.	0.	0.	0.	0.
Sys/SW Reqt	0.	0.	0.	0.	0.	0.	0.
SW Requirement	0.	0.	0.	0.	0.	0.	0.
Prelim Design	33.	7.	0.	0.	7.	7.	53.
Detail Design	33.	20.	0.	0.	7.	8.	68.
Code/Test	25.	23.	0.	0.	6.	7.	61.
CSCI Test	34.	43.	0.	0.	13.	15.	106.
System Test	0.	0.	0.	0.	0.	0.	0.
Oper T & E	0.	0.	0.	0.	0.	0.	0.
TOTAL	125.	94.	0.	0.	33.	37.	288.

SCHEDULE INFORMATION

Concept Start	JAN 87*	TRR	MAR 89* (4.3)
SRR	MAR 87* (3.7)	FCA	DEC 89* (9.1)
SDR	JUN 87* (2.5)	PCA	MAR 90* (2.8)
SSR	JAN 88 (7.1)	FQR	JUN 90* (2.8)
PDR	JUN 88* (4.6)	OTE	DEC 90* (6.9)
CDR	NOV 88* (5.0)		

SUPPLEMENTAL INFORMATION

Source Lines of Code	25000
Source Lines of Code/Person Month	86.7

CEI-3 Schedule and Cost Estimation System

MODERATE PROJECT
Current Input Parameters
Task: MODERATE

	Minimum	Nominal	Maximum	Mean	Std Dev
DEVELOPER TECHNOLOGY					
APPLication Complexity	3.0	3.0	3.0	3.0	0.0
Analyst CAPability	3.5	3.5	3.5	3.5	0.0
Application EXPerience	1.0	1.0	1.0	1.0	0.0
MODern Practices, use of	3.0	3.0	3.0	3.0	0.0
Programmer CAPability	3.5	3.5	3.5	3.5	0.0
TOOL support, automated	5.2	5.2	5.2	5.2	0.0
TURNaround, logon-hardcopy	2.0	2.0	2.0	2.0	0.0
ENVIRONMENTAL - COMPUTER					
DISP req'mt, special	6.8	6.8	6.8	6.8	0.0
MEMory Constraint	2.0	2.0	2.0	2.0	0.0
TIMe Constraint	6.0	6.0	6.0	6.0	0.0
Real TIMe	6.0	6.0	6.0	6.0	0.0
ENVIRONMENTAL - PRODUCT					
SPECification Level	6.0	6.0	6.0	6.0	0.0
QUALity Assurance Level	4.0	4.0	4.0	4.0	0.0
TEST Level	4.0	4.0	4.0	4.0	0.0
Requirements CHanGe vol	4.0	4.0	4.0	4.0	0.0
ReHOSTing develop->target	2.9	2.9	2.9	2.9	0.0
LANGuage type rating	1.0	1.0	1.0	1.0	0.0
Language EXPerience	1.0	1.0	1.0	1.0	0.0
SYSTem complexity-virtual	2.0	2.0	2.0	2.0	0.0
System EXPerience-virtual	1.0	1.0	1.0	1.0	0.0
Virtual Mach. VoLatility	0.0	0.0	0.0	0.0	0.0
ENVIRONMENTAL - SUPPORT					
MULTiple site development	0.0	0.0	0.0	0.0	0.0
Resource DEDication	10.0	10.0	10.0	10.0	0.0
Resource/support LOCation	0.0	0.0	0.0	0.0	0.0
SIZE & COMPLEXITY SUMMARY					
New Lines of Code	12500	12500	12500	12500	0
Existing Lines of Code	12500	12500	12500	12500	0
Lines to be Deleted	0	0	0	0	0
Lines to be Modified	0	0	0	0	0
Complexity	11.0	11.0	11.0	11.0	0.0

Current Input Parameters (Continued)

REUSE - REBUILD IMPACT

% Design Effort Needed . . .	50.0
% Implement Eff. Needed . .	50.0
% Testing Effort Needed . .	50.0

FINANCIAL FACTORS

Average Staff Pay Rate . . .	10000.0
Target Schedule	0.0
% Requirements Effort . . .	0.0
Req's Eff. Complete @ C/A .	0.0
Req's Schedule Constraint .	0.0
% Integration Effort	0.0
Avg. Annual Inflation . . .	0.0

MODERATE PROJECT
Summary Report
Task: MODERATE

(Minimum Time) Estimate

FULL SCALE DEVELOPMENT			
Development Time	21.14	0.00	Months
Development Effort	288.48	0.00	Person Months
Project Staff, Peak	20.70		Persons
Actual Staffing Rate	19.37	0.00	Persons/Year
Productivity	43.33		Lines/Staff/Month
Cost per Source Line	230.80		Dollars/Line
Total Full Scale Dev Task Cost	2884.95		K-Dollars
REQUIREMENTS and INTEGRATION			
Requirements Time	0.00		Months
Requirements Effort	0.00		Person Months
Total Requirments Cost	0.00		K-Dollars
Integration Time	0.00		Months
Integration Effort	0.00		Person Months
Total Integration Cost	0.01		K-Dollars
Complexity	11.00	0.00	
Basic Technology Rating	3666.94	0.00	
Effective Technology Rating	915.47	0.00	
Average Annual Inflation	0.00		
Effective Task Size	12500	0.0	Source Lines
Total Task Size	25000		Source Lines

MODERATE PROJECT
Milestone Report
Task: MODERATE

Milestone	Date (month)	Reqmts Effort (pmos)	Develop Effort (pmos)	Total Effort (pmos)
C/A	0.0	0.0	0.0	0.0
SDR	0.0	0.0	0.0	0.0
PDR	4.4	0.0	16.0	16.0
CDR	9.1	0.0	64.7	64.7
CUT	13.3	0.0	131.8	131.8
FQT	21.1	0.0	288.5	288.5
DT&E	21.1	0.0	288.5	288.5

Development Cost Spread Matrix
Allocated by Person Months
Part One
Task: MODERATE

Phase	System Eng.	Proj. Mgmt.	Design	Programmers
C/A	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0
SDR - PDR	0.0	0.1	14.9	0.6
PDR - CDR	0.0	1.9	38.1	4.2
CDR - CUT	0.0	3.2	7.5	39.4
CUT - FQT	0.0	12.4	23.9	44.4
SUBTOTAL	0.0	17.7	84.4	88.6
DT&E	0.0	0.0	0.0	0.0
Totals	0.0	17.7	84.4	88.6

Part Two

Phase	Q.A.	C.M.	Test	Data Manip.	Total
C/A	0.0	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0	0.0
SDR - PDR	0.1	0.1	0.1	0.1	16.0
PDR - CDR	1.0	0.8	1.4	1.3	48.8
CDR - CUT	3.2	2.4	8.5	2.8	67.0
CUT - FQT	8.8	8.8	55.7	2.7	156.7
SUBTOTAL	13.2	12.1	65.7	6.8	288.5
DT&E	0.0	0.0	0.0	0.0	0.0
Totals	13.2	12.1	65.7	6.8	288.5

Appendix E: PRICE S and System-3 Output for Complex Project

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

COMPLEX PROJECT — Development Item

ITEM DESCRIPTORS

Platform	1.8	Mgmt Complexity	1.20	Cost	0.0
Internal Integ	0.50	External Integ	0.50	Utilization	0.70

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Specification Review	188
Preliminary Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language JOVIAL		Source Code	40000	Non-executable SLOC	0.00
Complexity 1	1.20	Complexity 2	1.00	Productivity Factor	4.00

Application Categories	Mix	New Design	New Code
User Defined (APPL= 8.46)	1.00	0.70	0.70
Data S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Opr Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

COMPLEX PROJECT — Development Item

COSTS IN PERSON MONTHS

	Design	Prog	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	0.	0.	0.	0.	0.	0.	0.
Sys/SW Reqt	0.	0.	0.	0.	0.	0.	0.
SW Requirement	0.	0.	0.	0.	0.	0.	0.
Prelim Design	122.	27.	0.	0.	27.	28.	204.
Detail Design	126.	75.	0.	0.	30.	33.	264.
Code/Test	88.	79.	0.	0.	24.	29.	220.
CSCI Test	116.	144.	0.	0.	48.	58.	366.
System Test	0.	0.	0.	0.	0.	0.	0.
Oper T & E	0.	0.	0.	0.	0.	0.	0.
TOTAL	452.	325.	0.	0.	129.	148.	1055.

SCHEDULE INFORMATION

Concept Start	JAN 86*	TRR	APR 90* (8.0)
SRR	JUL 86* (7.2)	FCA	AUG 91* (16.1)
SDR	DEC 86* (4.8)	PCA	JAN 92* (5.2)
SSR	JAN 88 (13.2)	FQR	JUN 92* (5.2)
PDR	OCT 88* (8.9)	OTE	JUL 93* (12.9)
CDR	AUG 89* (9.7)		

SUPPLEMENTAL INFORMATION

Source Lines of Code	40000
Source Lines of Code/Person Month	37.9

CEI-3 Schedule and Cost Estimation System

COMPLEX PROJECT
Current Input Parameters
Task: COMPLEX

	Minimum	Nominal	Maximum	Mean	Std Dev
DEVELOPER TECHNOLOGY					
APPLication Complexity . . .	3.0	3.0	3.0	3.0	0.0
Analyst CAPability	5.5	5.5	5.5	5.5	0.0
Application EXPerience . . .	1.0	1.0	1.0	1.0	0.0
MODern Practices, use of . .	3.0	3.0	3.0	3.0	0.0
Programmer CAPability . . .	5.5	5.5	5.5	5.5	0.0
TOOL support, automated . .	5.2	5.2	5.2	5.2	0.0
TURNaround, logon-hardcopy .	2.0	2.0	2.0	2.0	0.0
ENVIRONMENTAL - COMPUTER					
DISP req'mt, special	6.8	6.8	6.8	6.8	0.0
MEMory Constraint	5.5	5.5	5.5	5.5	0.0
TIMe Constraint	7.0	7.0	7.0	7.0	0.0
Real TIMe	7.0	7.0	7.0	7.0	0.0
ENVIRONMENTAL - PRODUCT					
SPECification Level	6.0	6.0	6.0	6.0	0.0
QUALity Assurance Level . .	4.0	4.0	4.0	4.0	0.0
TEST Level	4.0	4.0	4.0	4.0	0.0
Requirements CHanGe vol . .	7.0	7.0	7.0	7.0	0.0
ReHOSTing develop->target .	5.1	5.1	5.1	5.1	0.0
LANGuage type rating	2.0	2.0	2.0	2.0	0.0
Language EXPerience	1.0	1.0	1.0	1.0	0.0
SYSTem complexity-virtual .	2.0	2.0	2.0	2.0	0.0
System EXPerience-virtual .	1.0	1.0	1.0	1.0	0.0
Virtual Mach. VoLatility . .	0.0	0.0	0.0	0.0	0.0
ENVIRONMENTAL - SUPPORT					
MULTiple site development .	6.5	6.5	6.5	6.5	0.0
Resource DEDication	10.0	10.0	10.0	10.0	0.0
Resource/support LOCation .	0.0	0.0	0.0	0.0	0.0
SIZE & COMPLEXITY SUMMARY					
New Lines of Code	28000	28000	28000	28000	0
Existing Lines of Code . . .	12000	12000	12000	12000	0
Lines to be Deleted	0	0	0	0	0
Lines to be Modified	0	0	0	0	0
Complexity	8.0	8.0	8.0	8.0	0.0

Current Input Parameters (Continued)

REUSE - REBUILD IMPACT

% Design Effort Needed . . .	70.0
% Implement Eff. Needed . .	70.0
% Testing Effort Needed . .	70.0

FINANCIAL FACTORS

Average Staff Pay Rate . . .	10000.0
Target Schedule	0.0
% Requirements Effort . . .	0.0
Req's Eff. Complete @ C/A .	0.0
Req's Schedule Constraint .	0.0
% Integration Effort	0.0
Avg. Annual Inflation . . .	0.0

COMPLEX PROJECT
Summary Report
Task: COMPLEX

(Minimum Time) Estimate

FULL SCALE DEVELOPMENT			
Development Time	36.21	0.00	Months
Development Effort	1055.46	0.00	Person Months
Project Staff, Peak	44.19		Persons
Actual Staffing Rate	24.14	0.00	Persons/Year
Productivity	26.53		Lines/Staff/Month
Cost per Source Line	376.97		Dollars/Line
Total Full Scale Dev Task Cost	10555.08		K-Dollars
REQUIREMENTS and INTEGRATION			
Requirements Time	0.00		Months
Requirements Effort	0.00		Person Months
Total Requirments Cost	0.00		K-Dollars
Integration Time	0.00		Months
Integration Effort	0.00		Person Months
Total Integration Cost	0.02		K-Dollars
Complexity	8.00	0.00	
Basic Technology Rating	4796.88	0.00	
Effective Technology Rating	625.69	0.00	
Average Annual Inflation	0.00		
Effective Task Size	28000	0.0	Source Lines
Total Task Size	40000		Source Lines

COMPLEX PROJECT
Milestone Report
Task: COMPLEX

Milestone	Date (month)	Reqmts Effort (pmos)	Develop Effort (pmos)	Total Effort (pmos)
C/A	0.0	0.0	0.0	0.0
SDR	0.0	0.0	0.0	0.0
PDR	7.6	0.0	58.5	58.5
CDR	15.6	0.0	236.9	236.9
CUT	22.8	0.0	482.2	482.2
FQT	36.2	0.0	1055.5	1055.5
DT&E	36.2	0.0	1055.5	1055.5

COMPLEX PROJECT
Development Cost Spread Matrix
Allocated by Person Months
Part One
Task: COMPLEX

Phase	System Eng.	Proj. Mgmt.	Design	Programmers
C/A	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0
SDR - PDR	0.0	0.5	54.6	2.3
PDR - CDR	0.0	7.0	139.4	15.5
CDR - CUT	0.0	11.8	27.4	144.0
CUT - FQT	0.0	45.3	87.4	162.4
SUBTOTAL	0.0	64.6	308.9	324.2
DT&E	0.0	0.0	0.0	0.0
Totals	0.0	64.6	308.9	324.2

Part Two

Phase	Q.A.	C.M.	Test	Data Manip.	Total
C/A	0.0	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0	0.0
SDR - PDR	0.3	0.2	0.3	0.3	58.5
PDR - CDR	3.8	2.8	5.2	4.7	178.4
CDR - CUT	11.8	8.9	31.0	10.3	245.3
CUT - FQT	32.3	32.3	203.8	9.7	573.3
SUBTOTAL	48.2	44.2	240.3	25.1	1055.5
DT&E	0.0	0.0	0.0	0.0	0.0
Totals	48.2	44.2	240.3	25.1	1055.5

Appendix F: PRICE S and System-3 Output for Very Complex Project

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

VERY COMPLEX PROJECT — Development Item

ITEM DESCRIPTORS

Platform	1.8	Mgmt Complexity	1.40	Cost	0.0
Internal Integ	0.50	External Integ	0.50	Utilization	0.80

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Specification Review	188
Preliminary Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language Ada		Source Code	60000	Non-executable SLOC	0.00
Complexity 1	1.30	Complexity 2	1.00	Productivity Factor	4.00

Application Categories	Mix	New Design	New Code
User Defined (APPL=10.95)	1.00	0.90	0.90
Data S/R	0.00	0.00	0.00
Online Comm	0.00	0.00	0.00
Realtime C&C	0.00	0.00	0.00
Interactive	0.00	0.00	0.00
Mathematical	0.00	0.00	0.00
String Manip	0.00	0.00	0.00
Opr Systems	0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

VERY COMPLEX PROJECT — Development Item

COSTS IN PERSON MONTHS

	Design	Prog	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	0.	0.	0.	0.	0.	0.	0.
Sys/SW Reqt	0.	0.	0.	0.	0.	0.	0.
SW Requirement	0.	0.	0.	0.	0.	0.	0.
Prelim Design	586.	133.	0.	0.	130.	140.	989.
Detail Design	594.	366.	0.	0.	152.	175.	1287.
Code/Test	361.	342.	0.	0.	119.	143.	965.
CSCI Test	410.	552.	0.	0.	227.	279.	1468.
System Test	0.	0.	0.	0.	0.	0.	0.
Oper T & E	0.	0.	0.	0.	0.	0.	0.
TOTAL	1951.	1392.	0.	0.	629.	737.	4709.

SCHEDULE INFORMATION

Concept Start	AUG 84*	TRR	JUN 92* (15.6)
SRR	JUL 85* (12.4)	FCA	JAN 95* (30.2)
SDR	MAR 86* (8.2)	PCA	NOV 95* (10.1)
SSR	JAN 88 (21.8)	FQR	SEP 96* (10.1)
PDR	JUL 89* (17.6)	OTE	OCT 98* (25.1)
CDR	MAR 91* (20.0)		

SUPPLEMENTAL INFORMATION

Source Lines of Code	60000
Source Lines of Code/Person Month	12.7

CEI-3 Schedule and Cost Estimation System

VERY COMPLEX PROJECT
Current Input Parameters
Task: VERYCOMP

	Minimum	Nominal	Maximum	Mean	Std Dev
DEVELOPER TECHNOLOGY					
APPLication Complexity . . .	3.0	3.0	3.0	3.0	0.0
Analyst CAPability	5.5	5.5	5.5	5.5	0.0
Application EXPerience . . .	1.0	1.0	1.0	1.0	0.0
MODern Practices, use of . .	5.2	5.2	5.2	5.2	0.0
Programmer CAPability . . .	5.5	5.5	5.5	5.5	0.0
TOOL support, automated . .	7.5	7.5	7.5	7.5	0.0
TURNaround, logon-hardcopy .	2.0	2.0	2.0	2.0	0.0
ENVIRONMENTAL - COMPUTER					
DISP req'mt, special	6.8	6.8	6.8	6.8	0.0
MEMory Constraint	6.0	6.0	6.0	6.0	0.0
TIMe Constraint	8.0	8.0	8.0	8.0	0.0
Real TIMe	8.0	8.0	8.0	8.0	0.0
ENVIRONMENTAL - PRODUCT					
SPECification Level	6.0	6.0	6.0	6.0	0.0
QUALity Assurance Level . .	4.0	4.0	4.0	4.0	0.0
TEST Level	4.0	4.0	4.0	4.0	0.0
Requirements CHanGe vol . .	10.0	10.0	10.0	10.0	0.0
ReHOSTing develop->target .	6.9	6.9	6.9	6.9	0.0
LANGuage type rating	3.0	3.0	3.0	3.0	0.0
Language EXPerience	0.5	0.5	0.5	0.5	0.0
SYSTem complexity-virtual .	3.0	3.0	3.0	3.0	0.0
System EXPerience-virtual .	0.5	0.5	0.5	0.5	0.0
Virtual Mach. VoLatility . .	0.0	0.0	0.0	0.0	0.0
ENVIRONMENTAL - SUPPORT					
MULTiple site development .	10.0	10.0	10.0	10.0	0.0
Resource DEDication	10.0	10.0	10.0	10.0	0.0
Resource/support LOCation .	0.0	0.0	0.0	0.0	0.0
SIZE & COMPLEXITY SUMMARY					
New Lines of Code	54000	54000	54000	54000	0
Existing Lines of Code . . .	6000	6000	6000	6000	0
Lines to be Deleted	0	0	0	0	0
Lines to be Modified	0	0	0	0	0
Complexity	7.0	7.0	7.0	7.0	0.0

Current Input Parameters (Continued)

REUSE - REBUILD IMPACT

% Design Effort Needed . . .	90.0
% Implement Eff. Needed . .	90.0
% Testing Effort Needed . .	90.0

FINANCIAL FACTORS

Average Staff Pay Rate . . .	10000.0
Target Schedule	0.0
% Requirements Effort . . .	0.0
Req's Eff. Complete @ C/A .	0.0
Req's Schedule Constraint .	0.0
% Integration Effort	0.0
Avg. Annual Inflation . . .	0.0

VERY COMPLEX PROJECT
Summary Report
Task: VERYCOMP

(Minimum Time) Estimate

FULL SCALE DEVELOPMENT			
Development Time	62.32	0.00	Months
Development Effort	4706.61	0.00	Person Months
Project Staff, Peak	114.51		Persons
Actual Staffing Rate	36.35	0.00	Persons/Year
Productivity	11.47		Lines/Staff/Month
Cost per Source Line	871.64		Dollars/Line
Total Full Scale Dev Task Cost	47068.44		K-Dollars
REQUIREMENTS and INTEGRATION			
Requirements Time	0.00		Months
Requirements Effort	0.00		Person Months
Total Requirments Cost	0.00		K-Dollars
Integration Time	0.00		Months
Integration Effort	0.00		Person Months
Total Integration Cost	0.08		K-Dollars
Complexity	7.00	0.00	
Basic Technology Rating	5504.40	0.00	
Effective Technology Rating	332.05	0.00	
Average Annual Inflation	0.00		
Effective Task Size	54000	0.0	Source Lines
Total Task Size	60000		Source Lines

VERY COMPLEX PROJECT
Milestone Report
Task: VERYCOMP

Milestone	Date (month)	Reqmts Effort (pmos)	Develop Effort (pmos)	Total Effort (pmos)
C/A	0.0	0.0	0.0	0.0
SDR	0.0	0.0	0.0	0.0
PDR	13.1	0.0	260.9	260.9
CDR	26.8	0.0	1056.3	1056.3
CUT	39.2	0.0	2150.1	2150.1
FQT	62.3	0.0	4706.8	4706.8
DT&E	62.3	0.0	4706.9	4706.9

VERY COMPLEX PROJECT
Development Cost Spread Matrix
Allocated by Person Months
Part One
Task: VERYCOMP

Phase	System Eng.	Proj. Mgmt.	Design	Programmers
C/A	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0
SDR - PDR	0.0	2.1	243.6	10.2
PDR - CDR	0.0	31.4	621.7	69.1
CDR - CUT	0.0	52.7	122.3	642.1
CUT - FQT	0.0	202.0	390.0	724.2
SUBTOTAL	0.0	288.2	1377.6	1445.6
DT&E	0.0	0.0	0.0	0.0
Totals	0.0	288.2	1377.6	1445.6

Part Two

Phase	Q.A.	C.M.	Test	Data Manip.	Total
C/A	0.0	0.0	0.0	0.0	0.0
C/A - SDR	0.0	0.0	0.0	0.0	0.0
SDR - PDR	1.1	0.9	1.6	1.4	260.9
PDR - CDR	16.7	12.6	23.0	20.9	795.5
CDR - CUT	52.7	39.5	138.3	46.1	1093.8
CUT - FQT	144.3	144.3	908.8	43.3	2556.7
SUBTOTAL	214.8	197.2	1071.7	111.7	4706.8
DT&E	0.0	0.0	0.0	0.0	0.0
Totals	214.8	197.2	1071.7	111.7	4706.8

Bibliography

1. Boehm, Barry W. Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall, Inc., 1981.

2. Brooks, Frederick P., Jr. The Mythical Man-Month: Essays on Software Engineering. Philippines: Addison-Wesley Publishing Company, Inc., 1975.

3. Clark, Major Gregory A. Software Cost Estimation Models -- Which One to Use? Student Report 86-0545. Air Command and Staff College (AU), Maxwell AFB AL, April 1986 (AD-B102-721).

4. Computer Economics, Inc. CEI Presents SYSTEM-3. Marina del Ray CA: Computer Economics, Inc., May 1988.

5. Department of the Air Force. Information Systems Program Management and Acquisition. AFR 700-4, Vol. II. Washington: HQ USAF, 15 March 1985.

6. Devenney, Capt Thomas J. An Exploratory Study of Software Cost Estimating at ESD. MS Thesis, GSM/SM/76S-4. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, July 1976 (AD-A030-162).

7. Ferens, Daniel V. Class handouts distributed in COST 672, Model Diagnostics and Software Management. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, February - March 1988.

8. Ferens, Daniel V. An Introduction To Software Parametric Cost Estimating. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, November 1987.

9. Fisher, Gene H. Cost Considerations in Systems Analysis. Contract DAHC-15-67-C-0150. New York NY: American Elsevier Publishing Company, Inc., 1971.

10. Frank, Paul M. Introduction to System Sensitivity Theory. New York NY: Academic Press, Inc., 1978.

11. Hollocker, Charles P. "Finding the Cost of Software Quality," IEEE Transactions on Engineering Management, EM-33: 223-228 (November 1986).

12. James, Lt Thomas G., Jr. Software Cost Estimating Methodology. Technical Report AFAL-TR-77-66. Dayton OH: Air Force Avionics Laboratory, August 1977 (AD-A048-192).

13. Kitchenham, Barbara A. and N.R. Taylor. "Software Project Development Cost Estimation," The Journal of Systems and Software, 5: 267-278 (1985).

14. Lamkey, Capt Robert J. and Curtis T. Pavy. Software Control During Development and Acquisition. MS Thesis, GLM. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 1980 (AD-A089-329).

15. Mitre Corporation. DOD Weapon System Acquisition and Management Study. Technical Report No. 6908, Vol. I. Bedford MA: Mitre Corporation, May 1975.

16. Otte, James E. Senior Member, Operations Staff, PRICE Systems. Personal Interview. Dayton OH. 23 July 1988.

17. Putnam, Lawrence H. Software Cost Estimating: A Quantitative Life Cycle Methodology (SLIM Reference Notebook). McLean VA: Quantitative Software Management, Inc., December 1981.

18. RCA Corporation. Price-S Reference Manual. Cherry Hill NJ: RCA Corp., October 1985.

19. RCA Corporation. Price-S Reference Manual. Cherry Hill NJ: RCA Corp., November 1987.

20. Schneider, Capt John, IV. A Preliminary Calibration of the RCA Price-S Software Cost Estimation Model. MS Thesis, GSM/SM/77S-15. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1977 (AD-A046-808).

21. Stanley, Wayne. Technical Support Staff, Computer Economics, Inc. Personal Interview. King of Prussia, PA. 28-29 June 1988.

22. Steffey, Capt Raymond E., Jr. An Analysis of the RCA Price-S Cost Estimation Model as it Relates to Current Air Force Computer Software Acquisition and Management. MS Thesis, GSM/SM/79D-20. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1979 (AD-A083-713).

23. Steig, Capt Jeffrey T. An Application of Discriminant Analysis to the Selection of Software Cost Estimating Models. MS Thesis, GSM/LSY/84S-26. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1984 (AD-A147-632).

24. Thibodeau, Robert. An Evaluation of Software Cost Estimating Models. RADC-TR-81-144. Los Angeles CA: General Research Corporation, June 1981 (AD-A104-226).

VITA

Captain Christina E. Voss [redacted] attended Ithaca College from which she received the degree of Bachelor of Arts in Mathematics in May 1983. Upon graduation, she was commissioned a Second Lieutenant in the United States Air Force through the AFROTC program at Cornell University. In October 1983, she was assigned to the Avionics Laboratory at Wright-Patterson AFB where she served as a Radar Systems Analyst, a Radar Development Project Manager, and Technical Manager, Radar System Design. Desiring a master's degree in cost analysis, she entered the School of Systems and Logistics, Air Force Institute of Technology, in June 1987.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GCA/LSQ/88D-11

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
School of Systems and Logistics

6b. OFFICE SYMBOL (If applicable)
LSQ

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS (City, State, and ZIP Code)
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433-6583

7b. ADDRESS (City, State, and ZIP Code)

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
See Box 19

12. PERSONAL AUTHOR(S)
Christina E. Voss, B.A., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1988 December	137

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Cost Estimates, Cost Models, Software Development Cost Estimates, Software Cost Estimating Models
12	05		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: A Sensitivity Analysis of the PRICE S and System-3 Software Cost Estimating Models

Thesis Chairman: Richard L. Murphy
Assistant Professor of Quantitative Management

Approved for public release IAW AFR 190-1.

WILLIAM A. MAUER
Associate Dean 27 JAN 1989
School of Systems and Logistics
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433-6583

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
Richard L. Murphy

22b. TELEPHONE (Include Area Code)
(513) 255-8410

22c. OFFICE SYMBOL
AFIT/LSQ

DD Form 1473, JUN 86 *Previous editions are obsolete.*

The purpose of this research effort was to perform an analysis on the PRICE S and System-3 software cost estimating models, in order to determine the domains of consistency for four software development efforts. That is, to determine the sets of inputs for which the two models provide approximately equal estimates.

It was necessary to examine the inputs required to run PRICE S, to determine reasonable values for those inputs to describe the four software development efforts of increasing complexity, and to run the PRICE S model to get the estimates for performing those efforts. The PRICE S estimates were used as the baseline for the sensitivity analyses on the System-3 variables.

The System-3 variables were examined and compared to the PRICE S variables. There were three types of relationships between the variables. There were some System-3 inputs that had one-to-one correspondence with PRICE S inputs, some that when combined with other System-3 inputs corresponded to a single PRICE S variable, and a few that had no PRICE S equivalent.

The sensitivity analysis on the System-3 variables was only applied to the ten variables that related to the PRICE S variable CPLX1 and the three variables that related to the PRICE S variable UTIL. The sensitivity analyses resulted in the determination of the domains of consistency for the four software efforts. Because of an infinite number of possible variable combinations, no one specific domain of consistency could be specified. Rather, the domains of consistency were represented by changes allowed in the variables, individually and as a group.